

1 Organization for the Advancement of Structured Information Systems

2 Business Transaction Protocol

3
4
5 An OASIS Committee Specification

6 ***CURRENT STATUS : internal committee draft***

7
8 Version 1.0 [0.9.0.3]

9 DD Mmm 2001 11 December 2001]

10
11

<i>Working draft 0.1 (pre-London)</i>	14 June 2001
<i>Working draft 0.2 (London)</i>	18 June 2001
<i>Working draft 0.3a (circulated)</i>	12 July 2001
<i>Working draft 0.3b (not circulated)</i>	17 July 2001
<i>Working draft 0.3c (circulated)</i>	20 July 2001
<i>Working draft 0.4 (circulated; incorporates PRF material)</i>	25 July 2001
<i>Working draft 0.5 (uncirculated)</i>	8 August 2001
<i>Working draft 0.6 (State tables)</i>	31 August 2001
<i>Working draft 0.7 (revised abs msgs) – (not circulated)</i>	28 September 2001
<i>Working draft 0.72 (completed abs msg revsn 0.7)</i>	3 October 2001
<i>Working draft 0.80 – full scope, PRF handback</i>	18 October 2001
<i>Working Draft 0.9</i>	24 October 2001
<i>Working Draft 0.9.0.1 – minor editorials issues applied</i>	16 November 2001
<i>Working Draft 0.9.0.2 – issue resolutions balloting to 10 Dec 2001</i>	4 December 2001
<i>Working Draft 0.9.0.3 – possible solution to msging issues</i>	11 December 2001

12
13 ***Change marks relative to 0.9.0.2 (i.e. assuming resolutions passed)***

14 Copyright and related notices

15
16 Copyright © The Organization for the Advancement of Structured Information Standards
17 (OASIS), 2001. All Rights Reserved.

18
19 This document and translations of it may be copied and furnished to others, and derivative
20 works that comment on or otherwise explain it or assist in its implementation may be
21 prepared, copied, published and distributed, in whole or in part, without restriction of any
22 kind, provided that the above copyright notice and this paragraph are included on all such
23 copies and derivative works. However, this document itself may not be modified in any way,
24 such as by removing the copyright notice or references to OASIS, except as needed for the
25 purpose of developing OASIS specifications, in which case the procedures for copyrights
26 defined in the OASIS Intellectual Property Rights document must be followed, or as required
27 to translate it into languages other than English.

28
29 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
30 successors or assigns.

31
32 This document and the information contained herein is provided on an "AS IS" basis and
33 OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
34 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION
35 HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF
36 MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

37
38
39 OASIS takes no position regarding the validity or scope of any intellectual property or other
40 rights that might be claimed to pertain to the implementation or use of the technology
41 described in this document or the extent to which any license under such rights might or
42 might not be available; neither does it represent that it has made any effort to identify any
43 such rights. Information on OASIS's procedures with respect to rights in OASIS
44 specifications can be found at the OASIS website. Copies of claims of rights made available
45 for publication and any assurances of licenses to be made available, or the result of an attempt
46 made to obtain a general license or permission for the use of such proprietary rights by
47 implementors or users of this specification, can be obtained from the OASIS Executive
48 Director.

49
50 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
51 applications, or other proprietary rights which may cover technology that may be required to
52 implement this specification. Please address the information to the OASIS Executive
53 Director.

54

54 **Acknowledgements**

55
56 Employees of the following companies participated in the finalization of this specification as
57 members of the OASIS Business Transactions Technical Committee:

58
59 BEA Systems, Inc.
60 Bowstreet, Inc.
61 Choreology Ltd.
62 Entrust, Inc.
63 Hewlett-Packard Co.
64 Interwoven Inc.
65 IONA Technologies PLC
66 SeeBeyond Inc.
67 Sun Microsystems Computer Corp.
68 Talking Blocks Inc.

69
70 The primary authors and editors of the main body of the specification were:

71
72 Alex Ceponkus (alex@ceponkus.org)
73 Peter Furniss (peter.furniss@choreology.com)
74 Alastair Green (alastair.green@choreology.com)
75

76 Additional contributions to its writing were made by

77
78 Sanjay Dalal (sanjay.dalal@bea.com)
79 Mark Little (mark_little@hp.com)
80

81 We thank Pal Takacsi-Nagy of BEA Systems Inc for his efforts in chairing the Technical
82 Committee, and Karl Best of OASIS for his guidance on the organization of the Committee's
83 work.
84
85

86
87 *In memory of Ed Felt*
88

89 Ed Felt of BEA Systems Inc. was an active and highly valued contributor to the work of the
90 OASIS Business Transactions Technical Committee.

91
92 His many years of design and implementation experience with the Tuxedo system,
93 Weblogic's Java transactions, and Weblogic Integration's Conversation Management
94 Protocol were brought to bear in his comments on and proposals for this specification.
95

96 He was killed in the crash of the hijacked United Airlines flight 93 near to Pittsburgh,
97 on 11 September 2001.
98

99 **Typographical and Linguistic Conventions and Style**

100
101 The initial letters of words in terms which are defined (at least in their substantive or
102 infinitive form) in the Glossary are capitalized whenever the term used with that exact
103 meaning, thus:

104
105 **Cancel**
106 **Participant**
107 **Application Message**
108

109 The first occurrence of a word defined in the Glossary is given in bold, thus:

110 **Coordinator**

111
112 Such words may be given in bold in other contexts (for example, in section headings or
113 captions) to emphasize their status as formally defined terms.
114
115

116 The names of abstract BTP protocol messages are given in upper-case throughout:

117
118 **BEGIN**
119 **CONTEXT**
120 **RESIGN**
121

122 The values of elements within a BTP protocol message are indicated thus:

123
124 **BEGIN/atom**
125

126 BTP protocol messages that are related semantically are joined by an ampersand:

127
128 **BEGIN/atom & CONTEXT**
129

130 BTP protocol messages that are transmitted together in a compound are joined by a + sign:

131
132 **ENROL + VOTE**
133

134 XML schemata and instances are given in Courier:

135
136 `<btpr:begin> ... </btpr:begin>`
137

138 Illustrative fragments of code in other languages, such as Java, are given in Lucida Console:

139
140 `int main (String[] args)`
141 `{`
142 `}`
143

144 Terms such as **MUST**, **MAY** and so on, which are defined in RFC [TBD number], “[TBD
145 title]” are used with the meanings given in that document but are given in lowercase bold,
146 rather than in upper-case:

147
148
149
150
151

An Inferior **must** send one of RESIGN, PREPARED or CANCELLED to its Superior.

151	Contents	
152		
153	Copyright and related notices	2
154	Acknowledgements	3
155	Typographical and Linguistic Conventions and Style	4
156	Contents	6
157	Part 1. Purpose and Features of BTP.....	9
158	Introduction.....	9
159	Development and Maintenance of the Specification	10
160	Overview of the Business Transaction Protocol.....	11
161	Part 2. Normative Specification of BTP.....	14
162	Actors, Roles and Relationships	14
163	Relationships	14
164	Roles involved in the Superior:Inferior relationship	16
165	Superior.....	16
166	Inferior	17
167	Enroller	18
168	Participant	19
169	Sub-coordinator.....	19
170	Sub-composer	20
171	Roles involved in the Terminator:Decider relationship	20
172	Decider.....	20
173	Coordinator.....	21
174	Composer	21
175	Terminator	21
176	Initiator.....	22
177	Factory	22
178	Other roles.....	23
179	Redirector.....	23
180	Status Requestor.....	23
181	Abstract Messages and Associated Contracts.....	24
182	Addresses	24
183	Request/response pairs.....	26
184	Compounding messages.....	26
185	Extensibility	28 27
186	Inferior handle	28
187	Messages	29 28
188	Qualifiers.....	29 28
189	CONTEXT	29
190	CONTEXT_REPLY.....	30
191	BEGIN	32 31
192	BEGUN.....	33 32
193	ENROL.....	34 33
194	ENROLLED	35 34
195	RESIGN	36 35
196	RESIGNED	37 36

197	PREPARE.....	37
198	PREPARED.....	39 38
199	CONFIRM.....	40
200	CONFIRMED.....	41 40
201	CANCEL.....	43 42
202	CANCELLED.....	44
203	CONFIRM_ONE_PHASE.....	46 45
204	HAZARD.....	47 46
205	CONTRADICTION.....	48 47
206	SUPERIOR_STATE.....	48
207	INFERIOR_STATE.....	50 49
208	REQUEST_CONFIRM.....	52 51
209	REQUEST_STATUSES.....	54 53
210	INFERIOR_STATUSES.....	54
211	REQUEST_STATUS.....	56 55
212	STATUS.....	57 56
213	REDIRECT.....	59 58
214	FAULT.....	60 59
215	Standard qualifiers.....	61
216	Transaction timelimit.....	65 64
217	Inferior timeout.....	65 64
218	Minimum inferior timeout.....	66 62
219	Inferior name.....	67 63
220	State Tables.....	69 65
221	Explanation of the state tables.....	69 65
222	Status queries.....	69 65
223	Decision events.....	69 65
224	Disruptions – failure events.....	70 66
225	Invalid cells and assumptions of the communication mechanism.....	70 66
226	Meaning of state table events.....	71 67
227	Persistent information.....	74 70
228	Failure Recovery.....	88 84
229	Types of failure.....	88 84
230	Persistent information.....	89 85
231	Redirection.....	90 86
232	Terminator:Decider failures.....	91 87
233	XML representation of Message Set.....	91 87
234	Addresses.....	91 87
235	Qualifiers.....	92 88
236	Identifiers.....	92 88
237	Message References.....	93 89
238	Messages.....	93 89
239	CONTEXT.....	93 89
240	CONTEXT -REPLY.....	93 89
241	BEGIN.....	93 89
242	BEGUN.....	94 90
243	ENROL.....	94 90
244	ENROLLED.....	94 90

245	RESIGN	9591
246	RESIGNED	9591
247	PREPARE.....	9591
248	PREPARED.....	9692
249	CONFIRM.....	9692
250	CONFIRMED.....	9692
251	CANCEL.....	9793
252	CANCELLED.....	9793
253	HAZARD.....	9894
254	CONTRADICTION.....	9894
255	SUPERIOR_STATE.....	9894
256	INFERIOR_STATE.....	9995
257	CONFIRM_ONE_PHASE.....	9995
258	REQUEST_CONFIRM.....	9995
259	REQUEST_STATUSES.....	10096
260	INFERIOR_STATUSES.....	10096
261	REQUEST_STATUS.....	10197
262	STATUS.....	10197
263	REDIRECT	10298
264	FAULT.....	10298
265	Standard qualifiers.....	10399
266	Transaction timelimit	10399
267	Inferior timeout	10399
268	Minimum inferior timeout.....	10399
269	Compounding of Messages.....	104100
270	Carrier Protocol Bindings	105101
271	Carrier Protocol Binding Proforma	105101
272	SOAP Binding	106102
273	Example scenario using SOAP binding	110104
274	SOAP + Attachments Binding.....	112106
275	XML Schema for SOAP Bindings	114107
276	Conformance	126120
277	Part 3. Appendices.....	128122
278	A. Glossary	128122
279		
280		

Part 1. Purpose and Features of BTP

Introduction

This document, which describes and defines the Business Transaction Protocol (BTP), is a Committee Specification of the Organization for the Advancement of Structured Information Standards (OASIS). The standard has been authored by the collective work of representatives of ten software product companies (listed on page 3), grouped in the Business Transactions Technical Committee (BT TC) of OASIS.

The OASIS BTP Technical Committee began its work at an inaugural meeting in San Jose, Calif. on 13 March 2001, and this specification was endorsed as a Committee Specification by a [*** unanimous] vote on [*** date].

BTP uses a two-phase outcome coordination protocol to create atomic effects (results of computations). BTP also permits the composition of such atomic units of work (atoms) into cohesive business transactions (cohesions) which allow application intervention into the selection of the atoms which will be confirmed, and of those which will be cancelled.

BTP is designed to allow transactional coordination of participants which are part of services offered by multiple autonomous organizations (as well as within a single organization). It is therefore ideally suited for use in a Web Services environment. For this reason this specification defines communications protocol bindings which target the emerging Web Services arena, while preserving the capacity to carry BTP messages over other communication protocols. Protocol message structure and content constraints are schematized in XML, and message content is encoded in XML instances.

The BTP allows great flexibility in the implementation of business transaction participants. Such participants enable the consistent reversal of the effects of atoms. BTP participants may use recorded before- or after-images, or compensation operations to provide the “roll-forward, roll-back” capacity which enables their subordination to the overall outcome of an atomic business transaction.

The BTP is an interoperation protocol which defines the roles which software agents (actors) may occupy, the messages that pass between such actors, and the obligations upon and commitments made by actors-in-roles. It does not define the programming interfaces to be used by application programmers to stimulate message flow or associated state changes.

The BTP is based on a permissive and minimal approach, where constraints on implementation choices are avoided. The protocol also tries to avoid unnecessary dependencies on other standards, with the aim of lowering the hurdle to implementation.

323 **Development and Maintenance of the Specification**

324
325 For more information on the genesis and development of BTP, please consult the OASIS BT
326 Technical Committee's website, at

327
328 <http://www.oasis-open.org/committees/business-transactions/>
329

330
331 As of the date of adoption of this specification the OASIS BT Technical Committee is still in
332 existence, with the charter of

- 333 maintaining the specification in the light of implementation experiences
- 334 coordinating publicity for BTP
- 335 liaising with other standards bodies whose work affects or may be affected by
336 BTP
- 337 reviewing the appropriate time, in the light of implementation experience and
338 user support, to put BTP forward for adoption as a full OASIS standard
- 339
- 340
- 341
- 342
- 343
- 344

345 If you have a question about the functionality of BTP, or wish to report an error or to suggest
346 a modification to the specification, please subscribe to:

347
348 bt-spec@lists.oasis-open.org
349

350 Any employee of a corporate member of OASIS, or any individual member of OASIS, may
351 subscribe to OASIS mail lists, and is also entitled to apply to join the Technical Committee.

352
353 The main list of the committee is:

354
355 business-transaction@lists.oasis-open.org
356
357
358
359
360
361

361 Overview of the Business Transaction Protocol

362 A Business Transaction is a consistent change in the state of a business relationship between
363 two or more parties. BTP provides means to allow the consistent and coordinated changes in
364 the relationship as viewed from each party.
365

366 BTP assumes that for a given business transaction state changes occur, or are desired, in some
367 set of parties, and that these changes are related in some business-defined manner.
368

369 Typically business-defined messages (“application messages”) are exchanged between the
370 parties to the transaction, which result in the performance of some set of operations. These
371 operations create provisional or tentative state changes (the transaction’s effect). The
372 provisional changes of each party must either be confirmed (given final effect), or must be
373 cancelled (counter-effected). Those parties which are confirmed create an atomic unit, within
374 which the business transaction should have a consistent final effect.
375

376 The meaning of “effect”, “final effect” and “counter-effect” is specific to each business
377 transaction and to each party’s role within it. A party may log intended changes (as its effect)
378 and only process them as visible state changes on confirmation (its final effect). Or it may
379 make visible state changes and store the information needed to cancel (its effect), and then
380 simply delete the information needed for cancellation (its final effect). A counter-effect may
381 be a precise inversion or removal of provisional changes, or it may be the processing of
382 operations that in some way compensate for, make good, alleviate or supplement their effect.
383
384

385 To ensure that confirmation or cancellation of the provisional effect within different parties
386 can be consistently performed, it is necessary that each party should
387

- 388 determine whether it is able both to cancel (counter-effect) and to confirm (give final
389 effect to) its effect
- 390 report its ability or inability to cancel-or-confirm (its preparedness) to a central
391 coordinating entity
392

393 After receiving these reports, the coordinating entity is responsible for determining which of
394 the parties should be instructed to confirm and which should be instructed to cancel.
395

396 Such a two-phase exchange (ask, instruct) mediated by a central coordinator is required to
397 achieve a consistent outcome for a set of operations. BTP defines the means for software
398 agents executing on network nodes to interoperate using a two-phase coordination protocol,
399 leading either to the abandonment of the entire attempted transaction, or to the selection of an
400 internally consistent set of confirmed operations.
401

402 BTP centres on the bilateral relationship between the computer systems of the coordinating
403 entity and those of one of the parties in the overall business transaction. In that relationship a
404 software agent within the coordinating entity’s systems plays the BTP role of Superior for a
405 given transaction and one or more software agents within the systems of the party play the
406 BTP role of Inferior. Each Inferior has one Superior, therefore, while a single Superior may
407

408 have multiple Inferiors within each party to the transaction, and may be related to Inferiors
409 within multiple parties. Each Superior:Inferior pair exchanges protocol-defined messages.
410
411 An Inferior is associated with some set of operation invocations that creates effect
412 (provisional or tentative changes) within the party, for a given business transaction. The
413 Inferior is responsible for reporting to its related Superior whether its associated operations'
414 effect can be confirmed/cancelled. A Superior is responsible for gathering the reports of all of
415 its Inferiors, in order to ascertain which should be cancelled or confirmed. For example, if a
416 Superior is acting as an atomic Coordinator it will treat any Inferior which cannot prepare to
417 cancel/confirm as having veto power over the whole business transaction, causing the
418 Superior to instruct all its Inferiors to cancel. A Superior may, under the dictates of a
419 controlling application, increase or reduce the set of Inferiors to which a common confirm or
420 cancel outcome may be delivered. Thus, the set of prepared Inferiors may be larger than the
421 set of confirmed Inferiors.
422
423 An Inferior:Superior relationship is typically established in relation to one or more
424 application messages sent from one part of the application (linked to the Superior) to some
425 other part of the application to request the performance of operations that are to be subject to
426 the confirm or cancel decision of the Superior. If an application is divided between a client
427 and a service, which use RPCs to communicate application requests and responses, then the
428 client would typically be associated with the Superior and the service would typically host the
429 Inferior(s). (BTP does not mandate such an application topology nor does it require the use of
430 RPC or any other application communication paradigm.)
431
432 BTP defines a CONTEXT message that can be sent "in relation to" such application
433 messages. On receipt of a CONTEXT, one or more Inferiors may be created and "enrolled"
434 with the Superior, establishing the Superior:Inferior relationships. The particular mechanisms
435 by which a CONTEXT is "related" to application messages is an issue for the application
436 protocol and its binding to carrier mechanisms. BTP does not require that the enrolment is
437 requested by any particular entity – in a particular implementation this may be done by the
438 Inferior itself, by parts of the application or by other entities involved in the transmission of
439 the CONTEXT and the application messages. BTP defines a CONTEXT_REPLY message
440 that can be sent on the return path of the CONTEXT to indicate whether the enrolment was
441 successful. Without CONTEXT_REPLY it would be possible for a Superior to have an
442 incorrect view of which Inferiors it was supposed to involve in its confirm decision.
443
444 It should be noted that this BTP specification recognises that:

- 445 ❑ an Inferior may itself be a Superior to other BTP Inferiors; this occurs when some of
446 the operations associated with the Inferior involve other application elements whose
447 operations are to be subject to the confirm/cancel instruction sent to the Inferior. The
448 specification treats any lower Inferiors as part of the associated operations;
- 449 ❑ the requirement on an Inferior to be able to confirm or cancel does not include any
450 specific mechanism to determine the isolation of the effects of operations; the
451 requirement is only that the Inferior is able to confirm or cancel the operations, as
452 their effects are known to the Superior and the application directly in contact with the
453 Superior. Thus the confirm-or-cancel requirement may be achieved by performing all
454 the operations and remembering a compensating counter operation (that will be

455 triggered by a cancel order); or by remembering the operations (having checked they
456 are valid) and performing them only if a confirm order is received; or by forbidding
457 any other access to data changed by the operations and releasing them in their
458 unchanged state (if cancelled) or their changed state (if confirmed); or by various
459 combinations of these. In addition, a cancellation may not return data to their original
460 state, but only to a state accepted by the application as appropriate to a cancelled
461 operation.
462
463
464
465
466
467
468

Part 2. Normative Specification of BTP

Actors, Roles and Relationships

Actors are software agents which process computations. BTP actors are addressable for the purposes of receiving application and BTP protocol messages transmitted over some underlying communications or carrier protocol. (See section “Addressing” for more detail.)

BTP actors play roles in the sending, receiving and processing of messages. These roles are associated with responsibilities or obligations under the terms of software contracts defined by this specification. (These contracts are stated formally in the sections entitled “Abstract Messages and Associated Contracts” and “State Tables”.) A BTP actor’s computations put the contracts into effect.

A role is defined and described in terms of a single business transaction. An implementation supporting a role may, as an addressable entity, play the same role in multiple business transactions, simultaneously or consecutively, or a separate addressable entity may be created for each transaction. This is a choice for the implementer, and the addressing mechanisms allow interoperation between implementations that make different choices.

Within a single transaction, one actor may play several roles, or each role may be assigned to a distinct actor. This is again a choice for the implementer. An actor playing a role is termed an “actor-in-role”.

Actors may interoperate, in the sense that the roles played by actors may be implemented using software created by different vendors for each actor-in-role. The section “Conformance”, gives guidelines on the groups of roles that may be implemented in a partial, interoperable implementation of BTP.

The descriptions of the roles concentrate on the normal progression of a business transaction, and some of the more important divergences from this. They do not cover all exception cases – the message set definition and the state tables provide a more comprehensive specification.

Note – A BTP role is approximately equivalent to an interface in some distributed computing mechanisms, or a port-type in WSDL. The definition of a role includes behaviour.

Relationships

There are two primary relationships in BTP.

- Between an application element that determines that a business transaction should be completed (the role of Terminator) and the BTP actor at the top of the transaction tree (the role of Decider);

511

512 □ Between BTP actors within the tree, where one (the Superior) will inform the other
513 (the Inferior) what the outcome decision is.

514

515 These primary relationships are involved in arriving at a decision on the outcome of a
516 business transaction, and propagating that decision to all parties to the transaction. Taking the
517 path that is followed when a business transaction is confirmed:

- 518 1. The Terminator determines that the business transaction should confirm, if it can; or
519 (for a Cohesion), which parts should confirm
- 520 2. The Terminator asks the Decider to apply the desired outcome to the tree, if it can
521 guarantee the consistency of the confirm decision
- 522 3. The Decider, which is Superior to one or more Inferiors, asks its Inferiors if they can
523 agree to a confirm decision (for a Cohesion, this may not be all the Inferiors)
- 524 4. If any of those Inferiors are also Superiors, they ask their Inferiors and so on down
525 the tree
- 526 5. Inferiors that are not Superiors report if they can agree to a confirm to their Superior
- 527 6. Inferiors that are also Superiors report their agreement only if they received such
528 agreement from their Inferiors, and can agree themselves
- 529 7. Eventually agreement (or not) is reported to the Decider. If all have agreed, the
530 Decider makes and persists the confirm decision (hence the term “Decider” – it
531 decides, everything else just asked); if any have disagreed, or if the confirm decision
532 cannot be persisted, a cancel decision is made
- 533 8. The Decider, as Superior tells its Inferiors of the outcome
- 534 9. Inferiors that are also Superiors tell their Inferiors, recursively down the tree
- 535 10. The Decider replies to the Terminator’s request to confirm, reporting the outcome
536 decision

537

538 There are other relationships that are secondary to Terminator:Decider, Superior:Inferior,
539 mostly involved in the establishment of the primary relationships.

540

541 The two primary relationships are linked in that a Decider is a Superior to one or more
542 Inferiors. There are also similarities in the semantics of some of the exchanges (messages)
543 within the relationships. However they differ in that

544

545 1. All exchanges between Terminator and Decider are initiated by the Terminator (it is
546 essentially a request/response relationship); either of Superior or Inferior may initiate
547 messages to the other

548

549 2. The Superior:Inferior relationship is recoverable – depending on the progress of the
550 relationship, the two sides will re-establish their shared state after failure; the
551 Terminator:Decider relationship is not recoverable

552

553 3. The nature of the Superior:Inferior relationship requires that the two parties know of
554 each other's addresses from when the relationship is established; the Decider does not
555 need to know the address of the Terminator (provided it has some way of returning
556 the response to a received message).
557

558 In the following sections, the responsibility of each role is defined, and the messages that are
559 sent or received by that role are listed. Note that some roles exist only to have a name for an
560 actor that issues a message and receives a reply to that message. Some of these roles may be
561 played by several actors in the course of a single business transaction.
562

563 **Roles involved in the Superior:Inferior relationship**

564 **Superior**

565
566
567 Accepts enrolments from Inferiors, establishing a Superior:Inferior relationship with each. In
568 cooperation with other actors and constrained by the messages exchanged with the Inferior,
569 the Superior determines the **Outcome** applicable to the Inferior and informs the Inferior by
570 sending CONFIRM or CANCEL. This outcome can be confirm only if a PREPARED
571 message is received from the Inferior, and if a record, identifying the Inferior can be
572 persisted. (Whether this record is also a record of a confirm decision depends on the
573 Superior's position in the business transaction as a whole.). The Superior must retain this
574 persistent record until it receives a CONFIRMED (or, in exceptional cases, CANCELLED or
575 HAZARD) from the Inferior.
576

577 A Superior may delegate the taking of the confirm or cancel decision to an Inferior, if there is
578 only one Inferior, by sending CONFIRM_ONE_PHASE.
579

580 A Superior may be *Atomic* or *Cohesive*; an Atomic Superior will apply the same decision to
581 all of its Inferiors; a Cohesive Superior may apply confirm to some Inferiors and cancel to
582 others, or may confirm some after others have reported cancellation. The set of Inferiors that
583 the Superior confirms (or attempts to confirm) is called the "confirm-set".
584

585 If RESIGN is received from an Inferior, the Superior:Inferior relationship is ended; the
586 Inferior has no further effect on the behaviour of the Superior as a whole.
587

588 A Superior receives

589
590 ENROL

591
592 to enrol a new Inferior, establishing a new Superior:Inferior relationship.
593

594 A Superior sends

595
596 ENROLLED

597
598 in reply to ENROL, if the appropriate parameter on the ENROL asked for the reply.
599

600 A Superior sends
601
602 PREPARE
603 CONFIRM
604 CANCEL
605 RESIGNED
606 CONFIRM_ONE_PHASE
607 SUPERIOR_STATE

608
609 to an enrolled Inferior.

610 A Superior receives
611
612
613 PREPARED
614 CANCELLED
615 CONFIRMED
616 HAZARD
617 RESIGN
618 INFERIOR_STATE

619
620 from an enrolled Inferior.

621 Inferior

622
623
624 Responsible for applying the Outcome to some set of associated operations – the application
625 determines which operations are the responsibility of a particular Inferior.

626
627 An Inferior is **Enrolled** with a single Superior (hereafter referred to as “its Superior”),
628 establishing a Superior:Inferior relationship. If the Inferior is able to ensure that either a
629 confirm or cancel decision can be applied to the associated operations, and can persist
630 information to retain that condition, it sends a PREPARED message to the Superior. When
631 the Outcome is received from the Superior, the Inferior applies it, deletes the persistent
632 information, and replies with CANCELLED or CONFIRMED as appropriate.

633
634 If an Inferior is unable to come to a prepared state, it cancels the associated operations and
635 informs the Superior with a CANCELLED message. If it is unable to either come to a
636 prepared state, or to cancel the associated operations, it informs the Superior with a
637 HAZARD message.

638
639 An Inferior that has become prepared may, exceptionally, make an autonomous decision to be
640 applied to the associated operations, without waiting for the Outcome from the Superior. It is
641 required to persist this autonomous decision and report it to the Superior with CONFIRMED
642 or CANCELLED as appropriate. If, when CONFIRM or CANCEL is received, the
643 autonomous decision and the decision received from the Superior are contradictory, the
644 Inferior must retain the record of the autonomous decision until receiving a
645 CONTRADICTION message.

646

647 An Inferior receives
648
649 PREPARE
650 CONFIRM
651 CANCEL
652 RESIGNED
653 CONFIRM_ONE_PHASE
654 SUPERIOR_STATE
655
656 from its Superior.
657
658 An Inferior sends
659
660 PREPARED
661 CANCELLED
662 CONFIRMED
663 HAZARD
664 RESIGN
665 INFERIOR_STATE
666
667 to its Superior.
668
669 An Inferior receives REQUEST_STATUS and replies with STATUS. If it is also a Superior,
670 the STATUS concerns the Inferior as a whole.
671
672 **Enroller**
673
674 Causes the enrolment of an Inferior with a Superior. This role is distinguished because in
675 some implementations the enrolment request will be performed by the application, in some
676 the application will ask the actor that will play the role of Inferior to enrol itself, and a
677 Factory may enrol a new Inferior (which will also be Superior) as a result of receiving
678 BEGIN&CONTEXT.
679
680 An Enroller sends
681
682 ENROL
683
684 to a Superior.
685
686 An Enroller receives
687
688 ENROLLED
689
690 in reply to ENROL if the Enroller asked for a response when the ENROL was sent.
691
692 An ENROL message sent from an Enroller that did not require an ENROLLED response may
693 be modified *en route* to the Superior by an intermediate actor to ask for an ENROLLED

694 response to be sent to the intermediate. (This may occur in the “one-shot” scenario, where an
695 ENROL/no-rsp-req is received in relation to a CONTEXT_REPLY/related; the receiver of
696 the CONTEXT_REPLY will need to ensure the enrolment is successful).
697

698 Participant

699
700 An Inferior which is specialized for the purposes of an application. Some application
701 operations are associated directly with the Participant, which is responsible for determining
702 whether a prepared condition is possible for them, and for applying the outcome. (“associated
703 directly” as opposed to involving another BTP Superior:Inferior relationship, in which this
704 actor is the Superior).

705
706 The associated operations may be performed by the actor that has the role of Participant, or
707 they may be performed by another actor, and only the confirm/cancel application is
708 performed by the Participant.
709

710 In either case, the Participant, as part of becoming prepared (i.e. before it can send
711 PREPARED to the Superior), will persist information allowing it apply a confirm decision to
712 the operations and to apply a cancel decision. The nature of this information depends on the
713 operations.

714 Note – Possible approaches are:

- 715 o The operations may be performed completely and the
716 Participant persists information to perform counter-effect
717 operations (compensating operations) to apply
718 cancellation;
 - 719 o The operations may be just checked and not performed at
720 all; the Participant persists information to perform them to
721 apply confirmation;
 - 722 o The Participants persists the prior state of data affected by
723 the operations and the operations are performed; the
724 Participant restores the prior state to apply cancellation;
 - 725 o As the previous, but other access to the affected data is
726 forbidden until the decision is known
-

727 Sub-coordinator

728
729 An Inferior which is also an Atomic Superior.
730

731
732 A sub-coordinator is the Inferior in one Superior:Inferior relationship and the Superior in one
733 or more Superior:Inferior relationships.
734

735 From the perspective of its Superior (the one the sub-coordinator is Inferior to), there is no
736 difference between a sub-coordinator and any other Inferior. From this perspective, the
737 “associated operations” of the sub-coordinator as an Inferior include the relationships with its
738 Inferiors.

739
740 A sub-coordinator does not become prepared (and send PREPARED to its Superior) until and
741 unless it has received PREPARED (or RESIGN) from all its Inferiors. The outcome is
742 propagated to all Inferiors.

743

744 **Sub-composer**

745

746 An Inferior which is also a Cohesive Superior.

747

748 Like a sub-coordinator, a sub-composer cannot be distinguished from any other Inferior from
749 the perspective of its Superior.

750

751 A sub-composer is similar to a sub-coordinator, except that the constraints linking the
752 different Inferiors concern only those Inferiors in the confirm-set. How the confirm-set is
753 controlled, and when, is not defined in this specification.

754

755 If the sub-composer is instructed to cancel, by receiving a CANCEL message from its
756 Superior, the cancellation is propagated to all its Inferiors.

757

758

759 **Roles involved in the Terminator:Decider relationship**

760

761 **Decider**

762

763 A Superior that is not the Inferior on a Superior:Inferior relationship. It is the top-node in the
764 transaction tree and receives requests from a Terminator as to the desired outcome for the
765 business transaction. If the Terminator asks the Decider to confirm the business transaction, it
766 is the responsibility of the Decider to finally take the confirm decision. The taking of the
767 decision is synonymous with the persisting of information identifying the Inferiors that are to
768 be confirmed. An Inferior cannot be confirmed unless PREPARED has been received from it.

769

770 A Decider is instructed to cancel by receiving CANCEL/whole.

771

772 A Decider that is an Atomic Superior (all Inferiors will have the same outcome) is a
773 Coordinator. A Decider that is a Cohesive Superior (some Inferiors may cancel, some
774 confirm) is a Cohesion.

775

776 All Deciders receive
777 REQUEST_CONFIRM
778 CANCEL/whole
779 REQUEST_STATUSES

780

781 All Deciders send

782 CONFIRMED
783 CANCELLED
784 INFERIOR_STATUSES
785
786 A Decider also receives REQUEST_STATUS and replies with STATUS, reporting its state
787 as a whole.
788
789 **Coordinator**
790
791 A Decider that is an Atomic Superior. The same outcome decision will be applied to all
792 Inferiors (excluding any from which RESIGN is received).
793
794 PREPARED must be received from all remaining Inferiors for a confirm decision to be taken.
795
796 A Coordinator must make a cancel decision if
797 it is instructed to cancel by the Terminator
798 if CANCELLED is received from any Inferior
799 if it is unable to persist a confirm decision
800
801 **Composer**
802
803 A Decider that is a Cohesive Superior. If the Terminator requests confirmation of the
804 Cohesion, that request will determine the confirm-set of the Cohesion.
805
806 PREPARED must be received from all Inferiors in the confirm-set (excluding any from
807 which RESIGN is received) for a confirm decision to be taken.
808
809 A Composer must make a cancel decision (applying to all Inferiors) if
810 it is instructed to cancel by the Terminator
811 if CANCELLED is received from any Inferior in the confirm-set
812 if it is unable to persist a confirm decision
813
814 A Composer may be asked to prepare some or all of its Inferiors by receiving PREPARE. It
815 issues PREPARE to any of those Inferiors from which none of PREPARED, CANCELLED
816 or RESIGNED have been received, and replies to the P REPARE with
817 INFERIOR_STATUSES.
818
819 A Composer may be asked to cancel some of its Inferiors, but not itself, by receiving
820 CANCEL/inferiors.
821
822 In addition to the messages received by the Composer as a Decider, it receives
823 PREPARE
824 CANCEL/inferiors
825
826 **Terminator**
827

828 Asks a Decider to confirm the business transaction, or instructs it to cancel all or (for a
829 Cohesion) part of the business transaction.
830
831 All communications between Terminator and Decider are initiated by the Terminator. A
832 Terminator is usually an application element.
833
834 A request to confirm is made by sending REQUEST_CONFIRM to the target Decider. If the
835 Decider is a Cohesion Composer, the Terminator may select which of the Composer's
836 Inferiors are to be included in the confirm-set. If the Decider is an Atom Coordinator, all
837 Inferiors are included. After applying the decision, the Decider replies with CONFIRMED,
838 CANCELLED or (in the case of problems) INFERIOR_STATUSES.
839
840 A Terminator may ask a Composer (but not a Coordinator) to prepare some or all of its
841 Inferiors with PREPARE/inferiors. The Composer replies with INFERIOR_STATUSES.
842
843 A Terminator may send CANCEL to instruct the Decider to cancel the whole business
844 transaction, or, if it is a Cohesion Composer, some of its Inferiors. The Decider replies with
845 CANCELLED, or for a selective cancel or in the case of problems, INFERIOR_STATUSES.
846
847 A Terminator may check the status of the Inferiors of the Decider by sending
848 REQUEST_STATUSES. The Decider replies with INFERIOR_STATUSES.
849
850 A Terminator sends
851 REQUEST_CONFIRM
852 CANCEL
853 PREPARE/inferiors
854 REQUEST_STATUSES
855
856 A Terminator receives
857 CONFIRMED
858 CANCELLED
859 INFERIOR_STATUSES
860
861 **Initiator**
862
863 Requests a **Factory** to create a Superior – this will either be a Decider (representing a new
864 top-level business transaction) or a sub-coordinator or sub-composer to be the Inferior of an
865 existing business transaction.
866
867 An Initiator sends
868
869 BEGIN
870 BEGIN & CONTEXT
871
872 to a Factory, and receives in reply
873
874 BEGUN & CONTEXT

875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921

Factory

Creates Superiors and returns the CONTEXT for the new Superior. The following types of Superior are created :

Decider, which is either
Composer or
Coordinator
Sub-composer
Sub-coordinator

A Factory receives

BEGIN
BEGIN & CONTEXT

and replies with

BEGUN & CONTEXT

If the BEGIN has no related CONTEXT, the Factory creates a Decider, either a Cohesion Composer or an Atom Coordinator, as determined by the “superior type” parameter on the BEGIN.

If the BEGIN has a related CONTEXT, the new Superior is also enrolled as an Inferior of the Superior identified by the CONTEXT. The new Superior is thus a sub-composer or sub-coordinator, as determined by the “superior type” parameter on the BEGIN.

Other roles

Redirector

Sends a REDIRECT message to inform any actor that an address previously supplied for some other actor is no longer appropriate, and to supply a new address or set of addresses to replace the old one.

A Redirector may send a REDIRECT message in response to receiving a message using the old address, or may send REDIRECT at its own initiative.

If a Superior moves from the superior-address in its CONTEXT, or an Inferior moves from the inferior-address in the ENROL message, the implementation **must** ensure that a Redirector catches any inbound messages using the old address and replies with a REDIRECT message giving the new address. (Note that the inbound message may itself be a REDIRECT message.)

922 A Redirector **may** also be used to change the address of other BTP actors.
923
924 After receiving a REDIRECT message, the BTP actor **must** use the new address not the old
925 one, unless failure prevents it updating its information.
926

927 **Status Requestor**

928
929 Requests and receives the current status of an Inferior or a Decider. The role of Status
930 Requestor has no responsibilities – it is just a name for where the REQUEST_STATUS
931 comes from.

932 A Status Requestor sends

933 REQUEST_STATUS

934 and receives

935 STATUS

936 in response.

937
938
939
940
941 The information returned will always relate to the actor concerned in its role as an Inferior,
942 even if it is also a Superior.
943
944
945

946 **Abstract Messages and Associated Contracts**

947
948 BT Protocol Messages are defined in this section in terms of the abstract information that has
949 to be communicated. These abstract messages will be mapped to concrete messages
950 communicated by a particular carrier protocol (there can be several such mappings defined).
951

952 The abstract message set and the associated state table assume the carrier protocol will

- 953 ❑ deliver messages completely and correctly, or not at all (corrupted messages will
954 not be delivered);
- 955 ❑ report some communication failures, but will not necessarily report all (i.e. not all
956 message deliveries are positively acknowledged within the carrier);
- 957 ❑ sometimes deliver successive messages in a different order than they were sent;
- 958 ❑
- 959 ❑
- 960 ❑

961 and

- 962 ❑ does not have built-in mechanisms to link a request and a response

963
964 Note that these assumptions would be met by a mapping to SMTP and more than met by
965 mappings to SOAP/HTTP.
966
967
968

969 However, when the abstract message set is mapped to a carrier protocol that provides a richer
970 service (e.g. reports all delivery failures, guarantees ordered delivery or offers a
971 request/response mechanism), the mapping can take advantage of these features. Typically in
972 such cases, some of the parameters of an abstract message will be implicit in the carrier
973 mechanisms, while the values of other parameters will be directly represented in transmitted
974 elements.
975
976

977 **Addresses**

978
979 All of the messages except CONTEXT and CONTEXT_REPLY have a “target address”
980 parameter and many also have other address parameters. These latter identify the desired
981 target of other messages in the set. In all cases, the exact value will invariably have been
982 originally determined by the implementation that is the target or desired future target.
983

984 The detailed format of the address will depend on the particular carrier protocol, but at this
985 abstract level is considered to have three parts. The first part, the “binding name”, identifies
986 the binding to a particular carrier protocol – some bindings are specified in this document,
987 others can be specified elsewhere. The second part of the address, the “binding address”, is
988 meaningful to the carrier protocol itself, which will use it for the communication (i.e. it will
989 permit a message to be delivered to a receiver). The third part, “additional information”, is
990 not used or understood by the carrier protocol. The “additional information” may be a
991 structured value.
992

993 When a message is actually transmitted, the “binding name” of the target address will identify
994 which carrier protocol is in use and the “binding address” will identify the destination, as
995 known to the carrier protocol. The entire binding address is considered to be “consumed” by
996 the carrier protocol implementation. All of it may be used by the sending implementation, or
997 some of it may be transmitted in headers, or as part of a URL in the carrier protocol, but then
998 used or consumed by the receiving implementation of the carrier protocol to direct the BTP
999 message to a BTP-aware entity (BTP-aware in that it is capable of interpreting the BTP
1000 messages). The “additional information” of the target address will be part of the BTP
1001 message itself and used in some way by the receiving BTP-aware entity (it could be used to
1002 route the message on to some other BTP entity). Thus, for the target address, only the
1003 “additional information” field is transmitted in the BTP message and the “additional
1004 information” is opaque to parties other than the recipient.
1005

1006 For other addresses in BTP messages, all three components will be within the message.
1007

1008 All messages that concern a particular Superior:Inferior relationship have an identifier
1009 parameter for the target side as well as the compound target address. This allows full
1010 flexibility for implementation choices – an implementation can:

- 1011
- 1012 a) Use the same binding address and additional information for multiple business
1013 transactions, using the identifier parameter to locate the relevant state
1014 information;

- 1015 b) Use the same binding address for multiple business transactions and use the
1016 additional information to locate the information; or
1017 c) Use a different binding address for each business transaction.
1018

1019 Which of these choices is used is opaque to the entity sending the message – both parts of the
1020 address and the identifier originated at the recipient of this message (and were transmitted as
1021 parameters of earlier messages in the opposite direction). In cases b) and c), the identifier is to
1022 some extent redundant, although interoperation requires that it always be present.
1023

1024 BTP recovery requires that the state information for a Superior or Inferior is accessible after
1025 failure and that the peer can distinguish between temporary inaccessibility and the permanent
1026 non-existence of the state information. As is explained in ‘Redirection’ below, BTP provides
1027 mechanisms – having a set of BTP addresses for some parameters, and the REDIRECT
1028 message – that make this possible, even if the recovered state information is on a different
1029 address to the original one (as may be the case if case c) above is used).
1030
1031

1032 **Request/response pairs**

1033

1034 Many of the messages combine in pairs as a request and its response. However, in some cases
1035 the response message is sent without a triggering request, or as a possible response to more
1036 than one type of request. To allow for this, the abstract message set treats each message as
1037 standalone; but where a request does expect a reply, a “reply-address” parameter will be
1038 present. For any message with a reply address parameter, in the case of certain errors, a
1039 FAULT message will be sent to the reply address instead of the expected reply.
1040

1041 For messages which are specified as sent between Superior and Inferior, a FAULT message is
1042 sent to the peer.
1043

1044 **Compounding messages**

1045

1046 BTP messages may be sent in combination with each other, or with other (application)
1047 messages. There are two cases:
1048

- 1049 a) Sending the messages together where the combination has semantic
1050 significance. One message is said to be “related to” the other – the combination
1051 is termed a “group”.
1052 b) Sending of the messages where the combination has no semantic significance,
1053 but is merely a convenience or optimisation. This is termed “bundling” – the
1054 combination is termed a “bundle”.
1055

1056 The form A&B is used to refer to a combination (group) where message B is sent in relation
1057 to A (“relation” is asymmetric). The form A+B is used to refer to A and B bundled together –
1058 the transmission of the bundle "A+B" is semantically identical to the transmission of A
1059 followed by the transmission of B.
1060

1061 Only certain combinations of messages are possible in a group, and the meaning of the
1062 relation is specifically defined for each such combination in the next section. A particular
1063 group is treated as a unit for transmission – it has a single target address. This is usually that
1064 of one of the messages in the group – the specification for the group defines which.

1065
1066 A “bundle” of messages may contain both unrelated messages and groups of related
1067 messages. The only constraint on which messages and groups can be bundled is that ~~in both~~
1068 ~~eases~~ the all messages will have the same binding address, but may have different “additional
1069 information” values. (Messages within a related group may have different addresses, where
1070 the rules of their relatedness permit this). Unless constrained by the binding, any messages or
1071 groups that are to be sent to the same binding address may be bundled – the fact that the
1072 binding addresses are the same is a necessary and sufficient condition for the sender to
1073 determine that the messages can be bundled.

1074
1075 A particular and important case of related messages is where a BTP CONTEXT message is
1076 sent related to an application message. In this case, the target of the application message
1077 defines the destination of the CONTEXT message. The receiving implementation may in fact
1078 remove the CONTEXT before delivering the application message to the application (Service)
1079 proper, but from the perspective of the sender, the two are sent to the same place.
1080 The compounding mechanisms, and the multi-part address structures, support the “one-wire”
1081 and “one-shot” communication patterns.

1082
1083 In “one-wire”, all message exchanges between two sides of a Superior:Inferior relationship,
1084 including the associated application messages, pass via the same “endpoints”. These
1085 “endpoints” may in fact be relays, routing messages on to particular actors within their
1086 domain. The onward routing will require some further addressing, but this has to be opaque to
1087 the sender. This can be achieved if the relaying endpoint ensures that all addresses for actors
1088 in its domain have the relay's address as their binding address, and any routing information it
1089 will need in its own domain is placed in the additional information. (This may involve the
1090 relay changing addresses in messages as they pass through it on the way out). On receiving a
1091 message, it determines the within-domain destination from the received additional
1092 information (which is thus rewritten) and forwards the message appropriately. The sender is
1093 unaware of this, and merely sees addresses with the same binding address, which it is
1094 permitted to bundle. The content of the “additional information” is a matter only for the relay
1095 – it could put an entire BTP address in there, or other implementation-defined information.
1096 Note that a quite different one-wire implementation can be constructed where there is no
1097 relaying, but the receiving entity effectively performs all roles, using the received identifiers
1098 to locate the appropriate state.

1099
1100 “One-shot” communication makes it possible to send an application message, receive the
1101 application reply, enrol an Inferior to be responsible for the confirm/cancel of the operations
1102 of those message and inform the Superior that the Inferior is prepared, all in one two-way
1103 exchange across the network (e.g. one request/reply of a carrier protocol). ~~concerns the~~
1104 ~~bundling of application messages, especially where the application uses a request/response~~
1105 ~~paradigm.~~ The application request is sent with a related CONTEXT message. The application
1106 response is sent with a ~~related~~ relation group of CONTEXT_REPLY/related, ~~with an~~
1107 ENROL/no-rsp-req message and a ~~bundled~~ PREPARED message ~~(assuming the operations~~
1108 ~~succeeded and the Inferior has decided to be prepared).~~ This is possible even if the Superior

1109 address is different the address of the application element that sends the original message (if
1110 the application exchange is request/reply, there may not even be an identifiable address for
1111 the application element). The target addresses of the ENROL and PREPARED (the Superior
1112 address) are not transmitted; the actor that was originally responsible for adding the
1113 CONTEXT to the outbound application message remembers the Superior address and
1114 forwards the ENROL and PREPARED appropriately. ~~must have a binding address that is the~~
1115 same as the target address of the application response (i.e. the reply address for the client, as
1116 perceived by the Service)—otherwise the Service cannot determine that it should bundle the
1117 messages together. One-shot is thus a specialization of one-wire.
1118

1119 With “one-shot”, if there are multiple Inferiors created as a result of a single application
1120 message, there is an ENROL and PREPARED message for each sent related to the
1121 CONTEXT_REPLY. ~~with the application response and the CONTEXT_REPLY.~~ If an
1122 operation fails, a CANCELLED message ~~can be~~ sent ~~with the response~~ instead of a
1123 PREPARED.
1124

1125 If the CONTEXT has “superior-type” of “atom”, then ~~if~~ subsequent messages to the same
1126 Service, with the same related CONTEXT/atom, can have their associated operations put
1127 under the control of the same Inferior, and only a CONTEXT_REPLY/completed is sent back
1128 with the response (if the new operations fail, it will be necessary to send back
1129 CONTEXT_REPLY/repudiated, or send CANCELLED). If the “superior type” on the
1130 CONTEXT is “cohesive”, each operation will require separate enrolment.
1131

1132 ~~Where does that last bit on one-shot, one-wire belong. It needs to be in somewhere.~~
1133 ~~prf~~
1134

1135 Extensibility

1136
1137 To simplify interoperation between implementations of this edition of BTP with
1138 implementations of future editions, the “must-be-understood” sub-parameter as specified for
1139 Qualifiers may be defined for use with any parameter added to an existing message in a future
1140 revision of this specification. The default for “must-be-understood” shall be “true”, so an
1141 implementation receiving an unrecognised parameter without a “false” value for “must-be-
1142 understood” shall not accept it (the FAULT value “UnrecognisedParameter” is available, but
1143 other errors, including lower-layer parsing/unmarshalling errors may be reported instead). If
1144 “must-be-understood” with the value “false” is present as a sub-parameter of a parameter in
1145 any message, a receiving implementation **should** ignore the parameter.
1146

1147 How the sub-parameter is associated with the new parameter is determined by the particular
1148 binding.
1149

1150 No special mechanism is provided to allow for the introduction of completely new messages.
1151

1152 Inferior handle

1153
1154 Some of the messages exchanged between a Terminator and a Decider are concerned with the
1155 individual Inferiors enrolled with the Decider, and not with the business transaction as a

1156 whole. These messages distinguish the Inferiors of Decider using an “inferior handle”. This is
1157 created by the Decider and is unambiguous within the scope of the Decider .

1158
1159 The “inferior handle” is distinct from the “inferior identifier” passed on an ENROL message
1160 (among other places). The latter is created by the Inferior (or its enroller) and is required to be
1161 unambiguous within the scope of the address-as-inferior on the ENROL (and unambiguous
1162 within **any** of the individual addresses in that set of BTP addresses - the identifier must
1163 identify the Inferior across all the places it might migrate to or that have recovery
1164 responsibility for it).

1165
1166 The “inferior handle” is only used by the Terminator to refer to the inferiors of the Decider.
1167 In messages between the Decider and its Inferiors, the address-as-inferior and inferior
1168 identifier are used.
1169

1170 Messages

1171 Qualifiers

1172
1173 All messages have a Qualifiers parameter which contains zero or more Qualifier values. A
1174 Qualifier has sub-parameters:
1175
1176

Sub-parameter	Type
qualifier name	string
qualifier group	URI
must-be-understood	Boolean
to-be-propagated	Boolean
content	Arbitrary – depends on type

1177
1178 **Qualifier group** ensures the Qualifier name is unambiguous. Qualifiers in the
1179 same group need not have any functional relationship. The qualifier group will
1180 typically be used to identify the specification that defines the qualifier’s meaning
1181 and use. Qualifiers may be defined in this or other standard specifications, in
1182 specifications of a particular community of users or of implementations or by
1183 bilateral agreement.
1184

1185 **Qualifier name** this identifies the meaning and use of the Qualifier, using a name
1186 that is unambiguous within the scope of the Qualifier group.

1187
1188 **Must-be-understood** if this has the value “true” and the receiving entity does
1189 not recognise the Qualifier type (or does not implement the necessary
1190 functionality), a FAULT “UnsupportedQualifier” shall be returned and the
1191 message shall not be processed. Default is “true”.
1192

1193 **To-be-propagated** if this has the value “true” and the receiving entity passes the
1194 BTP message (which may be a CONTEXT, but can be other messages) onwards
1195 to other entities, the same Qualifier value shall be included. If the value is
1196 “false”, the Qualifier shall not be automatically included if the BTP message is
1197 passed onwards. (If the receiving entity does support the qualifier type, it is
1198 possible a propagated message may contain another instance of the same type,
1199 even with the same Content – this is not considered propagation of the original
1200 qualifier.). Default is “false”.

1201
1202 **Content** the type (which may be structured) and meaning of the content is
1203 defined by the specification of the Qualifier.

1204 1205 1206 CONTEXT

1207
1208 A CONTEXT is supplied by (or on behalf of) a Superior and related to one or more
1209 application messages. (The means by which this relationship is represented is determined by
1210 the binding and the binding mechanisms of the application protocol.) The “superior type”
1211 parameter identifies whether the Superior will apply the same decision to all Inferiors
1212 enrolled using the same superior identifier (“superior type” is “atom”) or whether it may
1213 apply different decisions (“superior type” is “cohesion”).
1214

Parameter	Type
address-as-superior	Set of BTP addresses
superior identifier	Identifier
<u>reply address</u>	<u>BTP address</u>
superior type	cohesion/atom
qualifiers	List of qualifiers

1215
1216
1217 **address-as-superior** the address to which ENROL and other messages from an
1218 enrolled Inferior are to be sent. This can be a set of alternative addresses.

1219
1220 **superior identifier** identifies the Superior within the scope of the address-as-
1221 superior

1222
1223 reply address the address to which a replying CONTEXT REPLY is to be sent.
1224 This may be different each time the CONTEXT is transmitted – it refers to the
1225 destination of a replying CONTEXT REPLY for this particular transmission of
1226 the CONTEXT.

1227
1228 **superior type** identifies whether the CONTEXT refers to a Cohesion or an
1229 Atom. Default is atom.
1230

1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251

qualifiers standardised or other qualifiers. The standard qualifier “Transaction
timelimit” is carried by CONTEXT.

There is no target address parameter for CONTEXT as it is only transmitted in relation to the
application messages BEGIN and BEGUN.

The forms CONTEXT/cohesion and CONTEXT/atom refer to CONTEXT messages with the
superior type with the appropriate value.

CONTEXT_REPLY

CONTEXT_REPLY is sent after receipt of CONTEXT (related to application message(s)) to
indicate whether all necessary enrolments have already completed (ENROLLED has been
received) or will be completed by ENROL messages sent in relation to the
CONTEXT_REPLY or if an enrolment attempt has failed. CONTEXT_REPLY may be sent
related to an application message (typically the response to the application message related to
the CONTEXT). In some bindings the CONTEXT_REPLY may be implicit in the application
message.

Parameter	Type
<u>target-address</u>	<u>BTP address</u>
superior-address	BTP address
superior identifier	Identifier
completion_status	complete/related/repudiated
qualifiers	List of qualifiers

1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264

target-address the address to which the CONTEXT_REPLY is sent. This shall
be the “reply-address” from the CONTEXT.

superior-address one of the addresses from the address-as-superior from the
CONTEXT. (The parameter is present in CONTEXT_REPLY to disambiguate
the superior identifier.)

superior identifier the superior identifier from the CONTEXT

completion_status: reports whether all enrol operations made necessary by the
receipt of the earlier CONTEXT message have completed. Values are

value	meaning
<i>completed</i>	All enrolments (if any) have succeeded already
<i>related</i>	At least some enrolments are to be performed by ENROL messages related to the CONTEXT_REPLY. All

ENROL messages related to the CONTEXT_REPLY. All other enrolments (if any) have succeeded already.

repudiated

At least one enrolment has failed. The implications of receiving the CONTEXT have **not** been honoured.

1265

1266

qualifiers standardised or other qualifiers.

1267

1268

The form CONTEXT_REPLY/completed, CONTEXT_REPLY/related and CONTEXT_REPLY/repudiated refer to CONTEXT_REPLY messages with status having the appropriate value. The form CONTEXT_REPLY/ok refers to either of CONTEXT_REPLY/completed or CONTEXT_REPLY/related.

1269

1270

1271

1272

1273

If there are no necessary enrolments (e.g. the application messages related to the received CONTEXT did not require the enrolment of any Inferiors), then CONTEXT_REPLY/completed is used.

1274

1275

1276

1277

If a CONTEXT_REPLY/repudiated is received, the receiving implementation **must** ensure that the business transaction will not be confirmed.

1278

1279

1280

BEGIN

1281

1282

1283

A request to a Factory to create a new Business Transaction. This may either be a new top-level transaction, in which case the Composer or Coordinator will be the Decider, or the new Business Transaction may be immediately made the Inferior within an existing Business Transaction (thus creating a sub-Composer or sub-Coordinator).

1284

1285

1286

1287

Parameter	Type
target address	BTP address
reply address	BTP address
transaction type	cohesion/atom
qualifiers	List of qualifiers

1288

1289

target address the address of the entity to which the BEGIN is sent. How this address is acquired and the nature of the entity are outside the scope of this specification.

1290

1291

1292

1293

reply address the address to which the replying BEGUN and related CONTEXT message should be sent.

1294

1295

1296

transaction type identifies whether a new Cohesion or new Atom is to be created; this value will be the “superior type” in the new CONTEXT

1297

1298

1299 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction
1300 **timelimit**” may be present on BEGIN, to set the timelimit for the new business
1301 transaction and will be copied to the new CONTEXT. The standard qualifier
1302 “Inferior name” may be present if there is a CONTEXT related to the BEGIN.
1303

1304 A new top-level Business Transaction is created if there is no CONTEXT related to the
1305 BEGIN. A Business Transaction that is to be Inferior in an existing Business Transaction is
1306 created if the CONTEXT message for the existing Business Transaction is related to the
1307 BEGIN. In this case, the Factory is responsible for enrolling the new Composer or
1308 Coordinator as an Inferior of the Superior identified in that CONTEXT.
1309

1310 Note – This specification does not provide a standardised means to
1311 determine which of the Inferiors of a sub-Composer are in its confirm set.
1312 This is considered part of the application:inferior relationship.

1313 The forms BEGIN/cohesion and BEGIN/atom refer to BEGIN with “transaction type” having
1314 the corresponding value.
1315

1316 Types of FAULT possible (sent to Reply address)
1317

1318 General

1319 BEGUN

1320 BEGUN is a reply to BEGIN. There is always a related CONTEXT, which is the CONTEXT
1321 for the new business transaction.
1322

Parameter	Type
target address	BTP address
address-as-decider	Set of BTP addresses
transaction-identifier	Identifier
inferior-handle	Handle
address-as-inferior	Set of BTP addresses
qualifiers	List of qualifiers

1326 **target address** the address to which the BEGUN is sent. This will be the reply
1327 address from the BEGIN.
1328

1329 **address-as-decider** for a top-level transaction (no CONTEXT related to the
1330 BEGIN), this is the address to which PREPARE, REQUEST_CONFIRM,
1331 CANCEL and REQUEST_STATUS messages are to be sent; if a CONTEXT
1332 was related to the BEGIN this parameter is absent
1333

1334
1335 **transaction-identifier** identifies the new Composer or Coordinator within the
1336 scope of the address-as-decider. If this is not a top-level transaction, the
1337 transaction-identifier is optional, but if present shall be the inferior-identifier used
1338 in the enrolment with the Superior identified by the CONTEXT related to the
1339 BEGIN.

1340
1341 **inferior handle** Shall be absent if this is a top-level transaction and may or may
1342 not be present otherwise. (Presence or absence will be determined by the nature
1343 of the Superior identified in the CONTEXT related to the BEGIN). If present, the
1344 inferior handle will identify this new business transaction as in the inferiors-list
1345 parameters in messages between the Superior identified in the CONTEXT related
1346 to the BEGIN (acting as a Decider) and its Terminator. The value shall be
1347 different for each enrolled Inferior of that Superior.

1348
1349 **address-as-inferior** This parameter shall be absent if this is a top-level
1350 transaction and may be present, at implementation option otherwise. If present, it
1351 shall be the address-as-inferior used in the enrolment with the Superior identified
1352 by the CONTEXT related to the BEGIN. If this is a top-level transaction

1353
1354 **qualifiers** standardised or other qualifiers.

1355
1356 At implementation option, the “address-as-decider” and/or “address-as-inferior” and the
1357 “address-as-superior” in the related CONTEXT may be the same or may be different. There
1358 is no general requirement that they even use the same bindings. Any may also be the same as
1359 the target address of the BEGIN message (the inferior identifier on messages will ensure they
1360 are applied to the appropriate Composer or Coordinator).

1361
1362 No FAULT messages are issued on receiving BEGUN.

1363 1364 ENROL

1365
1366 A request to a Superior to ENROL an Inferior. This is typically issued after receipt of a
1367 CONTEXT message in relation to an application request.
1368 The actor issuing ENROL plays the role of Enroller.

1369

Parameter	type
target address	BTP address
superior identifier	Identifier
reply requested	Boolean
reply address	BTP address
address-as-inferior	Set of BTP addresses
inferior identifier	Identifier

Qualifiers

List of qualifiers

1370

1371

target address the address to which the ENROL is sent. This will be the address-as-superior from the CONTEXT message.

1372

1373

1374

superior identifier. The superior identifier as on the CONTEXT message

1375

1376

reply requested true if an ENROLLED response is required, false otherwise. Default is false.

1377

1378

1379

reply address the address to which a replying ENROLLED is to be sent, if “reply requested” is true. If this field is absent and “reply requested” is true, the ENROLLED should be sent to the “address-as-inferior” (or one of them, at sender’s option)

1380

1381

1382

1383

1384

address-as-inferior the address to which PREPARE, CONFIRM, CANCEL and SUPERIOR_STATE messages for this Inferior are to be sent.

1385

1386

1387

inferior identifier an identifier that unambiguously identifies this Inferior within the scope of any of the address-as-inferior set of BTP-addresses.

1388

1389

1390

qualifiers standardised or other qualifiers. The standard qualifier “Inferior name” may be present.

1391

1392

Types of FAULT possible (sent to Reply address)

1393

1394

1395

General

1396

InvalidSuperior – if superior identifier is unknown

1397

DuplicateInferior – if inferior with at least one of the set address-as-inferior the same and the same inferior identifier is already enrolled

1398

1399

WrongState – if it is too late to enrol new Inferiors (generally if the Superior has already sent a PREPARED message to its superior or terminator, or if it has already issued CONFIRM to other Inferiors).

1400

1401

1402

The form ENROL/rsp-req refers to an ENROL message with “reply requested” having the value “true”; ENROL/no-rsp-req refers to an ENROL message with “reply requested” having the value “false”

1403

1404

1405

1406

ENROL/no-rsp-req is typically sent in relation to CONTEXT_REPLY/related. ENROL/rsp-req is typically when CONTEXT_REPLY/completed will be used (after the ENROLLED message has been received.)

1407

1408

1409

1410

ENROLLED

1411

1412

Sent from Superior in reply to an ENROL/rsp-req message, to indicate the Inferior has been successfully enrolled (and will therefore be included in the termination exchanges)

1413

1414

1415

Parameter	Type
target address	BTP address
inferior identifier	Identifier
inferior-handle	Handle
Qualifiers	List of qualifiers

1416

1417

1418

1419

1420

target address the address to which the ENROLLED is sent. This will be the reply address from the ENROL message (or one of the address-as-inferiors if the reply address was empty)

1421

1422

inferior identifier The inferior identifier as on the ENROL message

1423

1424

1425

1426

1427

inferior handle the inferior handle that will identify this newly enrolled Inferior in the inferiors-list parameters in messages between the Superior (acting as a Decider) and its Terminator. This parameter is optional. The value shall be different for each enrolled Inferior of the Superior.

1428

1429

qualifiers standardised or other qualifiers.

1430

No FAULT messages are issued on receiving ENROLLED.

1431

1432

RESIGN

1433

1434

1435

1436

1437

1438

Sent from an enrolled Inferior to the Superior to remove the Inferior from the enrolment. This can only be sent if the operations of the business transaction have had no effect as perceived by the Inferior.

1439

1440

1441

1442

RESIGN may be sent at any time prior to the sending of a PREPARED or CANCELLED message (which cannot then be sent). RESIGN may be sent in response to a PREPARE message.

Parameter	type
target address	BTP address
superior identifier	identifier
address-as-inferior	Set of BTP addresses
inferior identifier	identifier
response requested	Boolean
Qualifiers	List of qualifiers

1443

1444 **target address** the address to which the RESIGN is sent. This will be the
 1445 superior address as used on the ENROL message.
 1446
 1447 **superior-identifier** The superior identifier as on the ENROL message
 1448
 1449 **address-as-inferior** The address-as-inferior as on the earlier ENROL message
 1450 (with the inferior identifier, this determines who the message is from)
 1451
 1452 **inferior-identifier** The inferior identifier as on the earlier ENROL message
 1453
 1454 **response-requested** is set to “true” if a RESIGNED response is required.
 1455
 1456 **qualifiers** standardised or other qualifiers.

1457 Note -- RESIGN is equivalent to readonly vote in some other protocols, but can be issued
 1458 early.
 1459

1460 Types of FAULT possible (sent to address-as-inferior)

1461 **General**

1462 **InvalidSuperior** – if superior identifier is unknown

1463 **InvalidInferior** – if no ENROL had been received for this address-as-
 1464 inferior and identifier (Inferior Identity)

1465 **WrongState** – if a PREPARED or CANCELLED has already been
 1466 received by the Superior from this Inferior
 1467

1468 The form RESIGN/rsp-req refers to an RESIGN message with “reply requested” having the
 1469 value “true”; RESIGN /no-rsp-req refers to an RESIGN message with “reply requested”
 1470 having the value “false”
 1471

1472 **RESIGNED**

1473 Sent in reply to a RESIGN/rsp-req message.
 1474

Parameter	Type
target address	BTP address
inferior identifier	Identifier
qualifiers	List of qualifiers

1475
 1476
 1477 **target address** the address to which the RESIGNED is sent. This will be the
 1478 address-as-inferior from the ENROL message.

1483 **inferior identifier** The inferior identifier as on the earlier ENROL message for
1484 this Inferior.

1485
1486 **qualifiers** standardised or other qualifiers.

1487
1488 After receiving this message the Inferior will not receive any more messages with this
1489 address-as-inferior and identifier.

1490
1491 No FAULT messages are issued on receiving RESIGNED.

1492
1493 **PREPARE**

1494
1495 Sent from Superior to an Inferior from whom ENROL but neither CANCELLED nor
1496 RESIGN have been received, requesting a PREPARED message. PREPARE can be sent after
1497 receiving a PREPARED message.

1498
1499 Sent from a Terminator to a Composer to tell it to prepare all or some of its inferiors, by
1500 sending PREPARE to any that have not already sent PREPARED, RESIGN or
1501 CANCELLED to the Composer. If the inferiors-list parameter is absent, the request applies to
1502 all the inferiors; if the parameter is present, it applies only to the identified inferiors of the
1503 Composer.

1504

Parameter	Type
target address	BTP address
inferior identifier	Identifier
reply address	BTP address
transaction-identifier	Identifier
inferiors-list	List of inferior handles
qualifiers	List of qualifiers

1505
1506 **target address** the address to which the PREPARE message is sent. When sent
1507 from Superior to Inferior, this will be the address-as-inferior from the ENROL
1508 message,. When sent from Terminator to Composer, this will be the decider-
1509 address from the BEGUN message .

1510
1511 **inferior identifier** When sent from Superior to Inferior, the inferior identifier as
1512 on the earlier ENROL message. This parameter shall be absent when sent from
1513 Terminator to Composer.

1514
1515 **reply address** When sent from Terminator to Composer, the address of the
1516 Terminator sending the PREPARE message. This parameter shall be absent when
1517 sent from Superior to Inferior.

1518

1519 **transaction identifier** When sent from Terminator to Composer, identifies the
1520 Composer and will be the transaction-identifier from the BEGUN message.. This
1521 parameter shall be absent when sent from Superior to Inferior.
1522

1523 **inferiors-list** When sent from Terminator to Composer, defines which of the
1524 Inferiors of this Composer preparation is requested for. If this parameter is absent
1525 when sent to a Composer, the PREPARE applies to all Inferiors. This parameter
1526 shall be absent when sent from Superior to Inferior.
1527

1528 **qualifiers** standardised or other qualifiers. The standard qualifier “Minimal
1529 inferior timeout” is carried by PREPARE.

1530
1531

1532 On receiving PREPARE, an Inferior **should** reply with a PREPARED, CANCELLED or
1533 RESIGN.

1534
1535

1536 When sent to a Composer, for all Inferiors identified in the inferiors-list parameter (all
1537 Inferiors if the parameter is absent), from which none of PREPARED, CANCELLED or
1538 RESIGNED has been received, the Composer shall issue PREPARE. It will reply to the
1539 Terminator, using the reply address on the PREPARE message, sending an
1540 INFERIOR_STATUSES message giving the status of the Inferiors identified on the inferiors-
1541 list parameter (all of them if the parameter was absent).

1542
1543

Types of FAULT possible (sent to Superior address)

1544
1545

General

1546
1547

UnknownTransaction – if the transaction-identifier is unknown

InvalidInferior – if inferior identifier is unknown, or an inferior-handle
on the inferiors-list is unknown

1548
1549

WrongState – if a CONFIRM or CANCEL has already been received by
this Inferior; if a REQUEST_CONFIRM or CANCEL/whole has already
been received by this Composer.

1550
1551

1552 The form PREPARE/whole refers to a PREPARE message sent to a Composer where the
1553 “inferiors-list” parameter is absent. The form PREPARE/inferiors refers to a PREPARE
1554 message sent to a Composer where the “inferiors-list” parameter is present. The unqualified
1555 form PREPARE is used for a PREPARE message sent to an Inferior.

1556
1557

PREPARED

1558
1559

1560 Sent from Inferior to Superior, either unsolicited or in response to PREPARE, but only when
1561 the Inferior has determined the operations associated with the Inferior can be confirmed and
1562 can be cancelled, as may be instructed by the Superior. The level of isolation is a local matter
1563 (i.e. it is the Inferiors choice, as constrained by the shared understanding of the application
1564 exchanges) – other access may be blocked, may see applied results of operations or may see
1565 the original state.

Parameter	Type
target address	BTP address
superior identifier	Identifier
address-as-inferior	Set of BTP addresses
inferior identifier	Identifier
default is cancel	Boolean
qualifiers	List of qualifiers

1566

1567

target address the address to which the PREPARED is sent. This will be the Superior address as on the ENROL message.

1568

1569

1570

superior identifier When the message is sent from an Inferior to the Superior, the superior identifier as on the ENROL message

1571

1572

1573

address-as-inferior When the message is sent from an Inferior to the Superior, the address-as-inferior as on the earlier ENROL message (with the inferior identifier, this determines who the message is from)

1574

1575

1576

inferior identifier The inferior identifier as on the ENROL message

1577

1578

1579

default is cancel if “true”, the Inferior states that if the outcome at the Superior is to cancel the operations associated with this Inferior, no further messages need be sent to the Inferior. If the Inferior does not receive a CONFIRM message, it will cancel the associated operations. The value “true” will invariably be used with a qualifier indicating under what circumstances (usually a timeout) an autonomous decision to cancel will be made. If “false”, the Inferior will expect a CONFIRM or CANCEL message as appropriate, even if qualifiers indicate that an autonomous decision will be made.

1580

1581

1582

1583

1584

1585

1586

1587

qualifiers standardised or other qualifiers. The standard qualifier “Inferior timeout” may be carried by PREPARED.

1588

1589

1590

On sending a PREPARED, the Inferior undertakes to maintain its ability to confirm or cancel the effects of the associated operations until it receives a CONFIRM or CANCEL message. Qualifiers may define a time limit or other constraints on this promise. The “default is cancel” parameter affects only the subsequent message exchanges and does not of itself state that cancellation will occur.

1591

1592

1593

1594

1595

1596

Types of FAULT possible (sent to address-as-inferior)

1597

1598

1599

General

InvalidSuperior – if Superior identifier is unknown

1600

1601 *InvalidInferior* – if no ENROL has been received for this address-as-
1602 inferior and identifier, or if RESIGN has been received from this Inferior

1603
1604 The form PREPARED/cancel refers to a PREPARED message with “default is cancel” =
1605 “true”. The unqualified form PREPARED refers to a PREPARED message with “default is
1606 cancel” = “false”.

1607
1608

1609 CONFIRM

1610

1611 Sent by the Superior to an Inferior from whom PREPARED has been received.

1612

Parameter	Type
target address	BTP address
inferior identifier	Identifier
qualifiers	List of qualifiers

1613

1614 **target address** the address to which the CONFIRM message is sent. This will
1615 be the address-as-inferior from the ENROL message.

1616

1617 **inferior identifier** The inferior identifier as on the earlier ENROL message for
1618 this Inferior.

1619

1620 **qualifiers** standardised or other qualifiers.

1621

1622 On receiving CONFIRM, the Inferior is released from its promise to be able to undo the
1623 operations of associated with the Inferior. The effects of the operations can be made available
1624 to everyone (if they weren't already).

1625

1626 Types of FAULT possible (sent to Superior address)

1627

1628 *General*

1629 *InvalidInferior* – if inferior identifier is unknown

1630 *WrongState* – if no PREPARED has been sent by, or if CANCEL has
1631 been received by this Inferior.

1632

1633

1634 CONFIRMED

1635

1636 Sent after the Inferior has applied the confirmation, both in reply to CONFIRM or when the
1637 Inferior has made an autonomous confirm decision, and in reply to a
1638 CONFIRM_ONE_PHASE if the Inferior decides to confirm its associated operations.

1639

1640 CONFIRMED is also sent by Decider to a Terminator in reply to REQUEST_CONFIRM if
 1641 all of the confirm-set confirms (and, for a Cohesion, all other Inferiors cancel) without
 1642 reporting hazards.
 1643

Parameter	Type
target address	BTP address
superior identifier	Identifier
address-as-inferior	Set of BTP addresses
inferior identifier	Identifier
address-as-decider	BTP address
transaction-identifier	identifier
confirm received	Boolean
qualifiers	List of qualifiers

1644
 1645 **target address** the address to which the CONFIRMED is sent. When sent by an
 1646 Inferior to a Superior, this will be the Superior address as on the CONTEXT
 1647 message. When sent from a Decider to a Terminator it will be the reply address
 1648 from the REQUEST_CONFIRM message.
 1649

1650 **superior identifier** When the message is sent from an Inferior to the Superior,
 1651 this shall be the superior identifier as on the CONTEXT message. This parameter
 1652 shall be absent when CONFIRMED is sent from Decider to Terminator.
 1653

1654 **address-as-inferior** When the message is sent from an Inferior to the Superior,
 1655 this shall be the address-as-inferior as on the earlier ENROL message (with the
 1656 inferior identifier, this determines who the message is from). This parameter shall
 1657 be absent when CONFIRMED is sent from Decider to Terminator.
 1658

1659 **inferior identifier** When the message is sent from an Inferior to the Superior, this
 1660 shall be the inferior identifier as on the earlier ENROL message. This parameter
 1661 shall be absent when CONFIRMED is sent from Decider to Terminator.
 1662

1663 **address-as-decider** When the message is sent from a Decider to the
 1664 Terminator, this shall be the address-as-decider of the Decider as on the BEGUN
 1665 message (with the transaction identifier, this determines who the message is
 1666 from). This parameter shall be absent when CONFIRMED is sent from an
 1667 Inferior to Superior.
 1668

1669 **transaction identifier** When the message is sent from a Decider to the
 1670 Terminator, this shall be the transaction identifier as on the BEGUN message (i.e.
 1671 the identifier of the Decider as a whole). This parameter shall be absent when
 1672 CONFIRMED is sent from an Inferior to Superior
 1673

1674 **confirm received** “true” if CONFIRMED is sent after receiving a CONFIRM
1675 message; “false” if an autonomous confirm decision has been made and either if
1676 no CONFIRM message has been received or the implementation cannot
1677 determine if CONFIRM has been received (due to loss of state information in a
1678 failure). This parameter shall be absent when CONFIRMED is sent from Decider
1679 to Terminator.

1680
1681 **qualifiers** standardised or other qualifiers.

1682
1683 Types of FAULT possible (sent to address-as-inferior)

1684
1685 **General**
1686 **InvalidSuperior** – if Superior identifier is unknown
1687 **InvalidInferior** – if no ENROL has been received for this address-as-
1688 inferior and identifier, or if RESIGN has been received from this Inferior.
1689

1690 Note – A CONFIRMED message arriving before a CONFIRM message is
1691 sent, or after a CANCEL has been sent will occur when the Inferior has
1692 taken an autonomous decision and is not regarded as occurring in the wrong
1693 state. (The latter will cause a CONTRADICTION message to be sent.)

1694 The form CONFIRMED/auto refers to a CONFIRMED message with “confirm
1695 received” = “false”; CONFIRMED/response refers to a CONFIRMED message
1696 with “confirm received” = “true”. The unqualified form CONFIRMED refers to
1697 the message without an confirm received parameter, as used between Decider
1698 and Terminator.
1699

1700
1701

1702 **CANCEL**
1703

1704 Sent by the Superior to an Inferior at any time before (and unless) CONFIRM has been sent.
1705

1706 Sent by a Terminator to a Decider at any time before REQUEST_CONFIRM has been sent.
1707

Parameter	Type
target address	BTP address
inferior identifier	Identifier
reply address	BTP address
transaction identifier	Identifier
inferiors-list	List of inferior handles
qualifiers	List of qualifiers

1708

1753 *InvalidInferior* – if inferior identifier is unknown, or an inferior-handle
1754 on the inferiors-list is unknown

1755 *WrongState* – if a CONFIRM has been received by this Inferior; if a
1756 REQUEST_CONFIRM has been received by this Composer.

1757
1758 The form CANCEL/whole refers to a CANCEL message sent to a Decider where the
1759 “inferiors-list” parameter is absent. The form CANCEL/inferiors refers to a CANCEL
1760 message sent to a Composer where the “inferiors-list” parameter is present. The unqualified
1761 form CANCEL is used to refer to a CANCEL message sent from a Superior to an Inferior.

1762
1763

1764 CANCELLED

1765

1766 Sent when the Inferior has applied (or is applying) cancellation of the operations associated
1767 with the Inferior. CANCELLED is sent from Inferior to Superior in the following cases:

1768

1769 1. before (and instead of) sending PREPARED, to indicate the Inferior is unable to
1770 apply the operations in full and is cancelling all of them;

1771

1772 2. in reply to CANCEL, regardless of whether PREPARED has been sent;

1773

1774 3. after sending PREPARED and then making and applying an autonomous
1775 decision to cancel.

1776

1777 4. in reply to CONFIRM_ONE_PHASE if the Inferior decides to cancel the
1778 associated operations

1779

1780 As is specified in the state tables, cases 1, 2 and 3 are not always distinct in some
1781 circumstances of recovery and resending of messages.

1782

1783 CANCELLED is also sent by Decider to a Terminator in reply to REQUEST_CONFIRM if
1784 all Inferiors cancel without reporting hazards.

1785

Parameter

target address	BTP address
superior identifier	Identifier
address-as-inferior	Set of BTP address
inferior identifier	Identifier
address-as-decider	BTP address
transaction-identifier	identifier
qualifiers	List of qualifiers

1786

1787 **target address** the address to which the CANCELLED is sent. When sent by an
1788 Inferior to a Superior, this will be the Superior address as on the CONTEXT
1789 message. When sent from a Decider to a Terminator it will be the reply address
1790 from the REQUEST_CONFIRM message.

1791
1792 **superior identifier** When the message is sent from an Inferior to the Superior,
1793 this shall be the superior identifier as on the CONTEXT message. This parameter
1794 shall be absent when CANCELLED is sent from Decider to Terminator.

1795
1796 **address-as-inferior** When the message is sent from an Inferior to the Superior,
1797 this shall be the address-as-inferior as on the earlier ENROL message (with the
1798 inferior identifier, this determines who the message is from). This parameter shall
1799 be absent when CANCELLED is sent from Decider to Terminator.

1800
1801 **inferior identifier** When the message is sent from an Inferior to the Superior, this
1802 shall be the inferior identifier as on the earlier ENROL message. This parameter
1803 shall be absent when CANCELLED is sent from Decider to Terminator.

1804
1805 **address-as-decider** When the message is sent from a Decider to the
1806 Terminator, this shall be the address-as-decider of the Decider as on the BEGUN
1807 message (with the transaction identifier, this determines who the message is
1808 from). This parameter shall be absent when CANCELLED is sent from an
1809 Inferior to Superior.

1810
1811 **transaction identifier** When the message is sent from a Decider to the
1812 Terminator, this shall be the transaction identifier as on the BEGUN message (i.e.
1813 the identifier of the Decider as a whole). This parameter shall be absent when
1814 CANCELLED is sent from an Inferior to Superior

1815
1816 **qualifiers** standardised or other qualifiers.

1817
1818 Types of FAULT possible (sent to address-as-inferior)

1819
1820 *General*

1821 *InvalidSuperior* – if Superior identifier is unknown

1822 *InvalidInferior* – if no ENROL has been received for this address-as-
1823 inferior and identifier, or if RESIGN has been received from this Inferior

1824 *WrongState* – if CONFIRM has been sent
1825

1826 Note – A CANCELLED message arriving before a CANCEL message is
1827 sent, or after a CONFIRM has been sent will occur when the Inferior has
1828 taken an autonomous decision and is not regarded as occurring in the wrong
1829 state. (The latter will cause a CONTRADICTION message to be sent.)

1830

1831

1832 CONFIRM_ONE_PHASE

1833
1834
1835
1836
1837

Sent from a Superior to an enrolled Inferior, when there is only one such enrolled Inferior. In this case the two-phase exchange is not performed between the Superior and Inferior and the outcome decision for the operations associated with the Inferior is determined by the Inferior.

Parameter	Type
target address	BTP address
inferior identifier	Identifier
report-hazard	boolean
qualifiers	List of qualifiers

1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852

target address the address to which the CONFIRM_ONE_PHASE message is sent This will be the address-as-inferior on the ENROL message.

inferior identifier The inferior identifier as on the earlier ENROL message for this Inferior.

report hazard Defines whether the superior wishes to be informed if a mixed condition occurs for the operations associated with the Inferior. If “report hazard” is “true”, the Inferior will reply with HAZARD if a mixed condition occurs, or if the Inferior cannot determine that a mixed condition has not occurred. If “report hazard” is false, the Inferior will report only its own decision, regardless of whether that decision was correctly and consistently applied. Default is false.

qualifiers standardised or other qualifiers.

1853
1854
1855
1856

CONFIRM_ONE_PHASE can be issued by a Superior to an Inferior from whom PREPARED has been received (subject to the requirement that there is only one enrolled Inferior).

1857
1858
1859

Types of FAULT possible (sent to Superior address)

General

InvalidInferior – if inferior identifier is unknown

WrongState – if a PREPARE has already been received from this Inferior

1864

HAZARD

1865
1866
1867
1868
1869
1870
1871

Sent when the Inferior has either discovered a “mixed” condition: that is unable to correctly and consistently cancel or confirm the operations in accord with the decision (either the received decision of the superior or its own autonomous decision), or when the Inferior is unable to determine that a “mixed” condition has not occurred.

1872 HAZARD is also used to reply to a CONFIRM_ONE_PHASE if the Inferior determines there
 1873 is a mixed condition within its associated operations or is unable to determine that there is not
 1874 a mixed condition.
 1875

Parameter	Type
target address	BTP address
superior identifier	Identifier
address-as-inferior	Set of BTP addresses
inferior identifier	Identifier
level	mixed/possible
Qualifiers	List of qualifiers

1876

1877 **target address** the address to which the MIXED is sent. This will be the
 1878 superior address from the ENROL message.

1879

1880 **superior identifier** The superior identifier as used on the ENROL message

1881

1882 **address-as-inferior** The address-as-inferior as on the earlier ENROL message
 1883 (with the inferior identifier, this determines who the message is from)

1884

1885 **inferior identifier** The inferior identifier as on the earlier ENROL message

1886

1887 **level** indicates, with value “mixed” that a mixed condition has definitely
 1888 occurred; or, with value “possible” that it is unable to determine whether a mixed
 1889 condition has occurred or not.

1890

1891 **qualifiers** standardised or other qualifiers.

1892

1893 Types of FAULT possible (sent to address-as-inferior)

1894

General

1895

InvalidSuperior – if Superior identifier is unknown

1896

InvalidInferior – if no ENROL has been received for this address-as-

1897

inferior and identifier, or if RESIGN has been received from this Inferior

1898

1899

1900

The form HAZARD/mixed refers to a HAZARD message with “level” = “mixed”, the form
 HAZARD/possible refers to a HAZARD message with “level” = “possible”.

1901

1902

1903

CONTRADICTION

1904

1905

Sent by the Superior to an Inferior that has taken an autonomous decision contrary to the
 decision for the atom. This is detected by the Superior when the ‘wrong’ one of

1906

1907

1908 CONFIRMED or CANCELLED is received. CONTRADICTION is also sent in response to a
1909 HAZARD message.
1910

Parameter	Type
target address	BTP address
inferior identifier	Identifier
Qualifiers	List of qualifiers

1911
1912 **target address** the address to which the CONTRADICTION message is sent.
1913 This will be the address-as-inferior from the ENROL message.
1914

1915 **inferior identifier** The inferior identifier as on the earlier ENROL message for
1916 this Inferior.
1917

1918 **qualifiers** standardised or other qualifiers.
1919

1920 Types of FAULT possible (sent to Superior address)
1921

1922 *General*

1923 *InvalidInferior* – if inferior identifier is unknown

1924 *WrongState* – if neither CONFIRMED or CANCELLED has been sent
1925 by this Inferior
1926

1927 SUPERIOR_STATE

1928
1929 Sent by a Superior as a query to an Inferior when
1930

- 1931 1. in the active state
- 1932
- 1933 2. there is uncertainty what state the Inferior has reached (due to recovery from
1934 previous failure or other reason).
1935

1936 Also sent by the Superior to the Inferior in response to a received INFERIOR_STATE, in
1937 particular states.
1938

Parameter	Type
target address	BTP address
inferior identifier	Identifier
Status	<i>see below</i>
reply requested	Boolean
Qualifiers	List of qualifiers

1939

1940 **target address** the address to which the SUPERIOR_STATE message is sent.
1941 This will be the address-as-inferior from the ENROL message.

1942
1943 **inferior identifier** The inferior identifier as on the earlier ENROL message for
1944 this Inferior.

1945
1946 **status** states the current state of the Superior, in terms of its relation to this
1947 Inferior only.

status value	meaning
<i>active</i>	The relationship with the Inferior is in the active state from the perspective of the Superior; ENROLLED has been sent, PREPARE has not been sent and PREPARED has not been received (as far as the Superior knows)
<i>prepared-received</i>	PREPARED has been received from the Inferior, but no outcome is yet available
<i>inaccessible</i>	The state information for the Superior, or for its relationship with this Inferior, if it exists, cannot be accessed at the moment. This should be a transient condition
<i>unknown</i>	The Inferior is not known – it does not exist from the perspective of the Superior. The Inferior can treat this as an instruction to cancel any associated operations

1949
1950 **Reply requested** true, if SUPERIOR_STATE is sent as a query at the Superior's
1951 initiative; false, if SUPERIOR_STATE is sent in reply to a received
1952 INFERIOR_STATE or other message. Can only be true if status is active or
1953 prepared-received.

1954
1955 **qualifiers** standardised or other qualifiers.

1956
1957 The Inferior, on receiving SUPERIOR_STATE with reply requested = true, should reply in a
1958 timely manner by (depending on its state) repeating the previous message it sent or by
1959 sending INFERIOR_STATE with the appropriate status value.

1960
1961 A status of unknown shall only be sent if it has been determined for certain that the Superior
1962 has no knowledge of the Inferior, or (equivalently) it can be determined that the relationship
1963 with the Inferior was cancelled. If there could be persistent information corresponding to the
1964 Superior, but it is not accessible from the entity receiving an INFERIOR_STATE/*y (or
1965 other) message targeted to the Superior or that entity cannot determine whether any such
1966 persistent information exists or not, the response shall be Inaccessible.

1967
1968 SUPERIOR_STATE/unknown is also used as a response to messages, other than
1969 INFERIOR_STATE/*y that are received when the Inferior is not known (and it is known
1970 there is no state information for it).
1971

1972 The form SUPERIOR_STATE/abcd refers to a SUPERIOR_STATE message status having a
 1973 value equivalent to “abcd” (for active, prepared-received, unknown and inaccessible) and
 1974 with “reply requested” = “false”. SUPERIOR_STATE/abcd/y refers to a similar message, but
 1975 with “reply requested” = “true”. The form SUPERIOR_STATE/*y refers to a
 1976 SUPERIOR_STATE message with “reply requested” = “true” and any value for status.
 1977
 1978

1979 **INFERIOR_STATE**
 1980

1981 Sent by an Inferior as a query when in the active state to a Superior, when (due recovery from
 1982 previous failure or other reason) there is uncertainty what state the Superior has reached.
 1983

1984 Also sent by the Inferior to the Superior in response to a received SUPERIOR_STATE, in
 1985 particular states.
 1986

Parameter	Type
target address	BTP address
superior identifier	Identifier
address-as-inferior	BTP address
inferior identifier	Identifier
Status	<i>see below</i>
reply requested	Boolean
Qualifiers	List of qualifiers

1987
 1988 **target address** the address to which the INFERIOR_STATE is sent. This will
 1989 be the target address as used the original ENROL message.
 1990

1991 **superior identifier** The superior identifier as used on the ENROL message
 1992

1993 **address-as-inferior** The address-as-inferior as on the ENROL message (with the
 1994 inferior identifier, this determines who the message is from)
 1995

1996 **inferior identifier** The inferior identifier as on the ENROL message
 1997

1998 **status** states the current state of the Inferior for the atomic business transaction,
 1999 which corresponds to the last message sent to the Superior by (or in the case of
 2000 ENROL for) the Inferior
 2001

status value	meaning/previous message sent
<i>active</i>	The relationship with the Superior is in the active state from the perspective of the Inferior; ENROL has been sent, a decision to send PREPARED has not been made.

<i>inaccessible</i>	The state information for the relationship with the Superior, if it exists, cannot be accessed at the moment. This should be a transient condition
<i>unknown</i>	The Inferior is not known – it does not exist from the perspective of the Superior. The Inferior can be treated as cancelled

2002

2003

2004

2005

2006

2007

2008

2009

reply requested “true” if INFERIOR_STATE is sent as a query at the Superior’s initiative; “false” if INFERIOR_STATE is sent in reply to a received SUPERIOR_STATE or other message. Can only be “true” if “status” is “active” or “prepared-received”. Can only be “true” if “status” is “active”.

qualifiers standardised or other qualifiers.

2010

2011

2012

2013

The Superior, on receiving INFERIOR_STATE with “reply requested” = “true”, should reply in a timely manner by (depending on its state) repeating the previous message it sent or by sending SUPERIOR_STATE with the appropriate status value.

2014

2015

2016

2017

2018

2019

2020

A status of “unknown” shall only be sent if it has been determined for certain that the Inferior has no knowledge of a relationship with the Superior. If there could be persistent information corresponding to the Superior, but it is not accessible from the entity receiving an SUPERIOR_STATE/*y (or other) message targetted on the Inferior or the entity cannot determine whether any such persistent information exists, the response shall be “inaccessible”.

2021

2022

2023

2024

INFERIOR_STATE/unknown is also used as a response to messages, other than SUPERIOR_STATE/*y that are received when the Inferior is not known (and it is known there is no state information for it).

2025

2026

2027

2028

2029

2030

2031

A SUPERIOR_STATE/INFERIOR_STATE exchange that determines that one or both sides are in the active state does not require that the Inferior be cancelled (unlike some other two-phase commit protocols). The relationship between Superior and Inferior, and related application elements may be continued, with new application messages carrying the same CONTEXT. Similarly, if the Inferior is prepared but the Superior is active, there is no required impact on the progression of the relationship between them.

2032

2033

2034

2035

2036

2037

The form INFERIOR_STATE/abcd refers to a INFERIOR_STATE message status having a value equivalent to “abcd” (for active, unknown and inaccessible) and with “reply requested” = “false”. INFERIOR_STATE/abcd/y refers to a similar message, but with “reply requested” = “true”. The form INFERIOR_STATE/*y refers to a INFERIOR_STATE message with “reply requested” = “true” and any value for status.

2038

2039

REQUEST_CONFIRM

2040

2041

2042

Sent from a Terminator to a Decider to request confirmation of the business transaction. If the business transaction is a Cohesion, the confirm-set is specified by the “inferiors-list” parameter.

2043

Parameter	Type
target address	BTP address
reply address	BTP address
transaction identifier	Identifier
inferiors-list	List of inferior handles
Report hazard	boolean
Qualifiers	List of qualifiers

2044

2045

2046

2047

target address the address to which the REQUEST_CONFIRM message is sent. This will be the address-as-decider on the BEGUN message.

2048

2049

reply address the address of the Terminator sending the REQUEST_CONFIRM message.

2050

2051

2052

2053

transaction identifier identifies the Decider. This will be the transaction-identifier from the BEGUN message.

2054

2055

2056

2057

inferiors-list defines which Inferiors enrolled with the Decider, if it is a Cohesion Composer, are to be confirmed. Shall be absent if the Decider is an Atom Coordinator.

2058

2059

2060

2061

2062

2063

2064

2065

report hazard Defines whether the Terminator wishes to be informed of hazard events and contradictory decisions within the business transaction. If “report hazard” is “true”, the receiver will wait until responses (CONFIRMED, CANCELLED or HAZARD) have been received from all of its inferiors, ensuring that any hazard events are reported. If “report hazard” is “false”, the Decider will reply with CONFIRMED or CANCELLED as soon as the decision for the transaction is known.

2066

2067

qualifiers standardised or other qualifiers.

2068

2069

2070

2071

If the “inferiors-list” parameter is present, the Inferiors identified shall be the “confirm-set” of the Cohesion. If the parameter is absent and the business transaction is a Cohesion, the “confirm-set” shall be all remaining Inferiors. If the business transaction is an Atom, the “confirm-set” is automatically all the Inferiors.

2072

2073

Any Inferiors from which RESIGN is received are not counted in the confirm-set.

2074

2075

2076

2077

If, for each of the Inferiors in the confirm-set, PREPARE has not been sent and PREPARED has not been received, PREPARE shall be issued to that Inferior.

2078
2079
2080
2081
2082

NOTE -- If PREPARE has been sent but PREPARED not yet received from an Inferior in the confirm-set, it is an implementation option whether and when to re-send PREPARE. The Superior implementation may choose to re-send PREPARE if there are indications that the earlier PREPARE was not delivered.

2083
2084
2085
2086
2087
2088
2089

A confirm decision may be made only if PREPARED has been received from all Inferiors in the “confirm-set”. The making of the decision shall be persistent (and if it is not possible to persist the decision, it is not made). If there is only one remaining Inferior in the “confirm set” and PREPARE has not been sent to it, CONFIRM_ONE_PHASE may be sent to it.

2090
2091

All remaining Inferiors that are not in the confirm set shall be cancelled.

2092
2093
2094

If a confirm decision is made and “report-hazard” was “false”, a CONFIRMED message shall be sent to the “reply-address”.

2095
2096
2097

If a cancel decision is made and “report-hazard” was “false”, a CANCELLED message shall be sent to the “reply-address”.

2098
2099
2100
2101

If “report-hazard” was “true” and any HAZARD or contradictory message was received (i.e. CANCELLED from an Inferior in the confirm-set or CONFIRMED from an Inferior not in the confirm-set), an INFERIOR_STATUSES reporting the status for all Inferiors shall be sent to the “reply-address”.

2102
2103
2104

Types of FAULT possible (sent to reply address)

2105
2106
2107
2108

General

UnknownTransaction – if the transaction-identifier is unknown

InvalidInferior – if an inferior handle in the inferiors-list is unknown

WrongState – if a CANCEL/whole has already been received .

2109
2110
2111
2112

The form REQUEST_CONFIRM/whole refers to a REQUEST_CONFIRM message where the “inferiors-list” parameter is absent. The form REQUEST_CONFIRM /inferiors refers to a REQUEST_CONFIRM message where the “inferiors-list” parameter is present.

2113
2114

REQUEST_STATUSES

2115
2116
2117
2118

Sent to a Decider to ask it to report the status of its Inferiors with an INFERIOR_STATUSES message.

Parameter	Type
target address	BTP address
reply address	BTP address

transaction identifier	Identifier
inferiors-list	List of inferior handles
Qualifiers	List of qualifiers

2119

2120

target address the address to which the REQUEST_STATUS message is sent..This will be the address-as-decider from the BEGUN message.

2121

2122

2123

reply address the address to which the replying INFERIOR_STATUSES is to be sent

2124

2125

2126

transaction identifier identifies the Decider. This will be the transaction-identifier from the BEGUN message.

2127

2128

2129

inferiors-list defines which inferiors enrolled with the Composer or Coordinator are to be included in the INFERIOR_STATUSES. If the list is absent, the status of all enrolled inferiors will be reported.

2130

2131

2132

qualifiers standardised or other qualifiers.

2133

2134

Types of FAULT possible (sent to reply-address)

2135

2136

General

2137

2138

The form REQUEST_STATUSES/whole refers to a REQUEST_STATUS with the inferiors-list absent. The form REQUEST_STATUS/inferiors refers to a REQUEST_STATUS with the inferiors-list present.

2139

2140

2141

2142

INFERIOR_STATUSES

2143

2144

Sent by a Decider to report the status of all or some of its inferiors in response to a REQUEST_STATUSES, PREPARE, CANCEL/inferiors and REQUEST_CONFIRM with "report-hazard" = "true".

2145

2146

2147

2148

Parameter	Type
target address	BTP address
address-as-decider	BTP address
transaction-identifier	identifier
status-list	Set of Status items - see below
general-qualifiers	List of qualifiers

2149

2150

target address the address to which the INFERIOR_STATUSES is sent. This will be the reply address on the received message

2151

2152

2153 **address-as-decider** The address-as-decider of the Decider as on the BEGUN
 2154 message (with the transaction identifier, this determines who the message is
 2155 from)
 2156

2157 **transaction identifier** The transaction identifier as on the BEGUN message (i.e.
 2158 the identifier of the Decider as a whole)
 2159

2160 **status-list** contains a number of Status-items, each reporting the status of one of
 2161 the inferiors of the Decider. The fields of a Status-item are
 2162

Field	Type
Inferior-handle	Inferior handle, identifying which inferior this Status-item contains information for.
status	One of the status values below (these are a subset of those for STATUS)
qualifiers	A list of qualifiers as received from the particular inferior or associated with the inferior in earlier messages (e.g. an Inferior name qualifier).

2163
 2164 The status value reports the current status of the particular inferior, as known to
 2165 the Composer or Coordinator. Values are:
 2166

status value	Meaning
<i>active</i>	The Inferior is enrolled
<i>resigned</i>	RESIGNED has been received from the Inferior
<i>preparing</i>	PREPARE has been sent to the inferior, none of PREPARED, RESIGNED, CANCELLED, HAZARD have been received
<i>prepared</i>	PREPARED has been received
<i>autonomously confirmed</i>	CONFIRMED/auto has been received, no completion message has been sent
<i>autonomously cancelled</i>	PREPARED had been received, and since then CANCELLED has been received but no completion message has been sent
<i>confirming</i>	CONFIRM has been sent, no outcome reply has been received
<i>confirmed</i>	CONFIRMED/response has been received
<i>cancelling</i>	CANCEL has been sent, no outcome reply has been received

status value	Meaning
<i>cancelled</i>	CANCELLED has been received, and PREPARED was not received previously
<i>cancel-contradiction</i>	Confirm had been ordered (and may have been sent), but CANCELLED was received
<i>confirm-contradiction</i>	Cancel had been ordered (and may have been sent) but CONFIRM/auto was received
<i>hazard</i>	A HAZARD message has been received

2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181

General qualifiers standardised or other qualifiers applying to the INFERIOR_STATUSES as a whole. Each Status-item contains a “qualifiers” field containing qualifiers applying to (and received from) the particular Inferior.

If the inferiors-list parameter was present on the received message, only the inferiors identified by that parameter shall have their status reported in status-list of this message. If the inferiors-list parameter was absent, the status of all enrolled inferiors shall be reported, except that an inferior that had been reported as *cancelled* or *resigned* on a previous INFERIOR_STATUSES message **may** be omitted (sender’s option).

REQUEST_STATUS

Sent to an Inferior or to a Decider to ask it to reply with STATUS.

Parameter	Type
target address	BTP address
reply address	BTP address
inferior identifier	Identifier
transaction-identifier	Identifier
Qualifiers	List of qualifiers

2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194

target address the address to which the REQUEST_STATUS message is sent. If the target is an Inferior, this will be the address-as-inferior on the ENROL message. If the target is a Decider, this will be the address-as-decider on the BEGUN message.

reply address the address to which the replying STATUS should be sent

inferior identifier If the target is an Inferior, the “inferior-identifier” on the ENROL message. If the target is a Decider, this parameter shall be absent.

transaction-identifier If the target is a Decider, the “transaction-identifier” on the BEGUN message. If the target is an Inferior, this parameter shall be absent.

2195
2196 **qualifiers** standardised or other qualifiers.

2197
2198 Types of FAULT possible (sent to reply address)

2199
2200 **General**

2201
2202
2203 **STATUS**

2204
2205 Sent by a Inferior or Decider in reply to a REQUEST_STATUS, reporting the overall state of
2206 the transaction tree node represented by the Inferior or Decider.

2207

Parameter	Type
target address	BTP address
address-as-inferior	BTP address
inferior identifier	Identifier
address-as-decider	BTP address
transaction-identifier	Identifier
status	See below
qualifiers	List of qualifiers

2208
2209 **target address** the address to which the STATUS is sent. This will be the reply
2210 address on the REQUEST_STATUS message

2211
2212 **address-as-inferior** If the sender is an Inferior, the address-as-inferior as on the
2213 ENROL message (with the inferior-identifier, this determines who the message is
2214 from). If the sender is a Decider, this parameter shall be absent

2215
2216 **inferior-identifier** If the sender is an Inferior, the inferior-identifier as on the
2217 ENROL message. If the sender is a Decider, this parameter shall be absent.

2218
2219 **address-as-decider** If the sender is a Decider, the address-as-decider on the
2220 BEGUN message (with the "transaction-identifier", this determines who the
2221 message is from). If the sender is an Inferior, this parameter shall be absent.

2222
2223 **transaction-identifier** If the sender is a Decider, the transaction identifier as on
2224 the BEGUN message. If the sender is an Inferior, this parameter shall be absent.

2225
2226 **status** states the current status of the transaction tree node represented by the
2227 sender.

2228

status value	Meaning from Inferior	Meaning from Decider
<i>Created</i>	The Inferior exists (and is addressable) but it has not been enrolled with a Superior	Not applicable
<i>Enrolling</i>	ENROL has been sent, but ENROLLED is awaited	Not applicable
<i>Active</i>	The Inferior is enrolled	New enrolment of inferiors is possible; no decision has been made.
<i>Resigning</i>	RESIGN has been sent; RESIGNED is awaited	Not applicable
<i>Resigned</i>	RESIGNED has been received	Not applicable
<i>Preparing</i>	PREPARE has been received; PREPARED has not been sent	Not applicable
<i>Prepared</i>	PREPARED has been sent; no outcome has been received or autonomous decision made	Not applicable
<i>Confirming</i>	CONFIRM has been received; CONFIRMED/response has not been sent	Confirm decision has been made but responses from inferiors are pending
<i>Confirmed</i>	CONFIRMED/response has been sent	CONFIRMED has been sent
<i>Cancelling</i>	CANCEL has been received or auto-cancel has been decided	Cancel decision has been made but responses from inferiors are pending
<i>Cancelled</i>	CANCELLED has been sent	CANCELLED has been sent
<i>cancel-contradiction</i>	Autonomous cancel decision was made, CONFIRM received; CONTRADICTION has not been received	Not applicable
<i>confirm-contradiction</i>	Autonomous confirm decision was made, CANCEL received; CONTRADICTION has not been received	Not applicable
<i>Hazard</i>	A hazard has been discovered; CONTRADICTION has not been received	A hazard has been reported from at least one Inferior
<i>Contradicted</i>	CONTRADICTION has been received	Not applicable
<i>Unknown</i>	No state information for the	No state information for the

status value	Meaning from Inferior	Meaning from Decider
	identifier exists; no such Inferior exists	transaction identifier exists; no such Decider exists
<i>Inaccessible</i>	There may be state information for this identifier but it cannot be reached/existence cannot be determined	There may be state information for this identifier but it cannot be reached/existence cannot be determined

2229
2230
2231
2232
2233
2234
2235
2236
2237

qualifiers standardised or other qualifiers.

REDIRECT

Sent when the address previously given for a Superior or Inferior is no longer valid and the relevant state information is now accessible with a different address (but the same superior or inferior identifier).

Parameter	Type
target address	BTP address
superior identifier	Identifier
inferior identifier	Identifier
old address	Set of BTP addresses
new address	Set of BTP addresses
qualifiers	List of qualifiers

2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256

target address the address to which the REDIRECT is sent. This may be the reply address from a received message or the address of the opposite side (superior/inferior) as given in a CONTEXT or ENROL message

superior identifier The superior identifier as on the CONTEXT message and used on an ENROL message. (present only if the REDIRECT is sent from the Inferior).

inferior identifier The inferior identifier as on the ENROL message

old address The previous address of the sender of REDIRECT. A match is considered to apply if any of the old addresses match one that is already known.

new address The (set of alternatives) new addresses to be used for messages sent to this entity.

qualifiers standardised or other qualifiers.

2257 If the actor whose address is changed is an Inferior, the new address value
 2258 replaces the address-as-inferior as present in the ENROL.
 2259
 2260 If the actor whose address is changed is a Superior, the new address value
 2261 replaces the Superior address as present in the CONTEXT message (or as present
 2262 in any other mechanism used to establish the Superior:Inferior relationship).
 2263
 2264

2265 **FAULT**

2266 Sent in reply to various messages to report an error condition
 2267
 2268

Parameter	Type
target address	BTP address
superior identifier	Identifier
inferior identifier	Identifier
fault type	See below
fault data	See below
qualifiers	List of qualifiers

2269
 2270 **target address** the address to which the FAULT is sent. This may be the reply
 2271 address from a received message or the address of the opposite side
 2272 (superior/inferior) as given in a CONTEXT or ENROL message
 2273

2274 **superior identifier** the superior identifier as on the CONTEXT message and as
 2275 used on the ENROL message (present only if the FAULT is sent to the superior).
 2276

2277 **inferior identifier** the inferior identifier as on the ENROL message (present only
 2278 if the FAULT is sent to the inferior)
 2279

2280 **fault type** identifies the nature of the error, as specified for each of the main
 2281 messages.
 2282

2283 **fault data** information relevant to the particular error. Each fault type defines the
 2284 content of the fault data:
 2285

fault type	meaning	fault data
<i>General</i>	Any otherwise unspecified problem	Free text explanation
<i>UnknownParameter</i>	A BTP message has been received with an unrecognised parameter	Free text explanation

<i>WrongState</i>	The message has arrived when the recipient is in an invalid state.	
<i>CommunicationFailure</i>	Any fault arising from the carrier mechanism and communication infrastructure.	Determined by the carrier mechanism and binding specification
<i>InvalidSuperior</i>	The received identifier is not known or does not identify a known Superior	The identifier
<i>DuplicateInferior</i>	An inferior with the same address and identifier is already enrolled with this Superior	The identifier
<i>InvalidInferior</i>	The Superior is known but the Inferior identified by the address-as-inferior and identifier are not enrolled in it	The Inferior Identity (address-as-inferior and identifier)
<i>UnsupportedQualifier</i>	A qualifier has been received that is not recognised and on which "must-be-Understood" is "true".	Qualifier group and name

2286
2287
2288

qualifiers standardised or other qualifiers.

2289
2290
2291

Note – If the carrier mechanism used for the transmission of BTP messages is capable of delivering messages in a different order than they were sent in, the "WrongState" FAULT is not sent and should be ignored if received.

2292

Groups – combinations of related messages

2293
2294

The following combinations of messages form related groups, for which the meaning of the group is not just the aggregate of the meanings of the messages. The "&" notation is used to indicate relatedness.

2295
2296
2297
2298

CONTEXT & application message

2299
2300

Meaning: the application work implied by the application message is to be part of the business transaction identified by the CONTEXT. Changes in application state resulting from the application message are to be subject to the outcome delivered to an enrolled Inferior – this requires the enrolment of a new Inferior if no appropriate Inferior is enrolled of if the CONTEXT is for cohesion.

2301
2302
2303
2304
2305
2306

2307 Target address: the target address is that of the application message. It is not required
2308 that the application address be a BTP address (in particular, there is no BTP-defined
2309 “additional information” field – the application protocol (and its binding) may or may not
2310 have a similar construct).

2311
2312 There may be multiple application messages related to a single CONTEXT message. All
2313 the application messages are part of the business transaction identified by the CONTEXT.
2314 This specification does not imply any further relatedness among the application messages
2315 themselves (though the application might).

2316
2317 The actor that sends the group shall retain knowledge of the Superior address in the
2318 CONTEXT. If the CONTEXT is a CONTEXT/atom, the actor shall also keep track of
2319 transmitted CONTEXTs for which no CONTEXT_REPLY has been received.

2320
2321 If the CONTEXT is a CONTEXT/atom, the actor receiving the CONTEXT shall ensure
2322 that a CONTEXT_REPLY message is sent back to the reply address of the CONTEXT
2323 with the appropriate completion status.

2324

2325 Note – The representation of the relation between CONTEXT and one or
2326 more application messages depends on the binding to the carrier protocol. It
2327 is not necessary that the CONTEXT and application messages be closely
2328 associated “on the wire” (or even sent on the same connection) – some kind
2329 of referencing mechanism may be used.

2330

CONTEXT_REPLY and ENROL/no-rsp-req

2331

2332

2333

2334

2335

2336

2337

2338

2339

2340

2341

2342

2343

2344

2345

2346

2347

2348

2349

2350

2351

2352

Meaning: the enrolment of the Inferior identified in the ENROL is to be performed with
the Superior identified in the CONTEXT message this CONTEXT_REPLY is replying
to. If the “completion-status” of CONTEXT_REPLY is “related”, failure of this
enrolment shall prevent the confirmation of the business transaction.

Target address: the target address is that of the CONTEXT_REPLY. This will be the
reply address of the CONTEXT message (in many cases, including request/reply
application exchanges, this address will usually be implicit).

The target address of the ENROL message is omitted.

The actor receiving the related group will use the retained Superior address from the
CONTEXT sent earlier to forward the ENROL. When doing so, it changes the ENROL to
ask for a response and supplies its own address as the reply address. If this attempt fails
(i.e. ENROLLED is not received), and the “completion-status” of the
CONTEXT_REPLY was “related”, the actor is required to ensure that the Superior does
not proceed to confirmation. How this is achieved is an implementation option, but must
take account of the possibility that direct communication with the Superior may fail. (One
method is to prevent REQUEST_CONFIRM being sent to the Superior (in its role as
Decider); another is to enrol as another Inferior before sending the original CONTEXT

2353 out with an application message). If the Superior is a sub-coordinator or sub-composer,
2354 an enrolment failure must ensure the sub-coordinator does not send PREPARED to its
2355 own Superior.

2356
2357 If the actor receiving the related group is also the Superior (i.e. it has the same binding
2358 address), the explicit forwarding of the ENROL is not required, but the resultant effect –
2359 that either the enrolment worked or the Superior does not confirm – shall be the same.

2360
2361 A CONTEXT_REPLY & ENROL/no-rsp-req group may contain multiple ENROL
2362 messages, for several Inferiors. Each ENROL shall be forwarded and an ENROLLED
2363 reply received before the Superior is allowed to confirm (if the “completion-status” is
2364 “related”).

2365
2366 If the CONTEXT had “superior-type” value of “atom”, the “completion-status” of the
2367 CONTEXT_REPLY shall be “related”. The “superior-type” was “cohesive”, the
2368 “completion-status” shall be “completed” or “related”. If the value is “completed”, the
2369 actor receiving the group shall forward the ENROLS, but is not required to (though it
2370 may) prevent confirmation.

2371 2372 CONTEXT_REPLY & ENROL/no-rsp-req & PREPARED & CANCELLED

2373
2374 This combination is considered to apply if either, PREPARED or CANCELLED are
2375 related to CONTEXT_REPLY, with or without ENROL/no-rsp-req. It is central to the
2376 “one-shot” mechanism.

2377
2378 Meaning: If ENROL is present, the meaning and required processing is the same as for
2379 CONTEXT_REPLY & ENROL/no-rsp-req. The PREPARED or CANCELLED
2380 message(s) are forwarded to the Superior identified in the CONTEXT message this
2381 CONTEXT_REPLY is replying to.

2382
2383 Target address: the target address is that of the CONTEXT_REPLY. This will be the
2384 reply address of the CONTEXT message (in many cases, including request/reply
2385 application exchanges, this address will usually be implicit).

2386
2387 The target address of the PREPARED and CANCELLED message is omitted.

2388
2389 The actor receiving the group forwards the PREPARED or CANCELLED message to the
2390 Superior in exactly the same as for an ENROL, using the retained Superior address from
2391 the CONTEXT sent earlier, except there is no reply required from the Superior.

2392
2393 If (as is usual) an ENROL and PREPARED or CANCELLED message are for the same
2394 Inferior, the ENROL shall be sent first, but the actor need not wait for the ENROLLED to
2395 come back before sending the PREPARED or CANCELLED (so an
2396 ENROL+PREPARED bundle from this actor to the Superior could be used).

2397
2398 The group can contain multiple ENROL, PREPARED and CANCELLED messages.
2399 Each PREPARED and CANCELLED message will be for a different Inferior.. There is

2400 no constraint on the order of their forwarding, except that ENROL and PREPARED or
2401 CANCELLED for the same Inferior shall be delivered to the Superior in the order
2402 ENROL first, followed by the other message for that Inferior..
2403
2404
2405

2406 CONTEXT_REPLY & ENROL & application message (& PREPARED)

2407
2408 Meaning: the relation between the BTP messages is as for the preceding groups and the
2409 application work implied by the application message has been associated with the Inferior
2410 identified by the ENROL and will be subject to the outcome delivered to that Inferior.
2411

2412 Target address: the target address of the group is the target address of the
2413 CONTEXT_REPLY which shall also be the target address of the application message.
2414 The ENROL and PREPARED messages do not contain their target addresses.
2415

2416 The processing of the ENROL and PREPARED messages is the same as for the previous
2417 groups.

2418 Should this be possible with ENROL/rsp-req ? That means the
2419 interceptor/communicator has to remember the enroller address

2420
2421 This group is used when participation in business transaction (normally a cohesion), is
2422 initiated by the service (Inferior) side, which fetches or acquires the CONTEXT, with
2423 some associated application semantic, performs some work for the transaction and sends
2424 an application message with a related ENROL. The CONTEXT_REPLY allows the
2425 addressing of the application (and the CONTEXT_REPLY) to be distinct from that of the
2426 Superior.
2427

2428 The actor receiving the group may associate the “inferior-handle” received on the
2429 ENROLLED with the application message in a manner that is visible to the application
2430 receiving the message.
2431

2432 BEGUN & CONTEXT

2433
2434 Meaning: the CONTEXT is that for the new business transaction, containing the
2435 Superior address.
2436

2437 Target address: the target address is that of the BEGUN message – this will be the reply
2438 address of the earlier BEGIN message.
2439

2440 BEGIN & CONTEXT

2441
2442 Meaning: the new business transaction is to be an Inferior (sub-coordinator or sub-
2443 composer) of the Superior identified by the CONTEXT. The Factory (receiver of the
2444 BEGIN) will perform the enrolment.
2445

2446 Target address: the target address is that of the BEGIN – this will be the address of the
2447 Factory.

2448

2449 **Standard qualifiers**

2450

2451 The following qualifiers are expected to be of general use to many applications and
2452 environments. The URI “urn:oasis:names:tc:BTP:qualifiers” is used in the
2453 Qualifier group value for the qualifiers defined here.

2454

2455

2456 **Transaction timelimit**

2457

2458 The transaction timelimit allows the Superior (or an application element initiating the
2459 business transaction) to indicate the expected length of the active phase, and thus give an
2460 indication to the Inferior of when it would be appropriate to initiate cancellation if the active
2461 phase appears to continue too long. The time limit ends (the clock stops) when the Inferior
2462 decides to be prepared and issues PREPARED to the Superior.

2463

2464 It should be noted that the expiry of the time limit does not change the permissible actions of
2465 the Inferior. At any time prior to deciding to be prepared (for an Inferior), the Inferior is
2466 **permitted** to initiate cancellation for internal reasons. The timelimit gives an indication to the
2467 entity of when it will be useful to exercise this right.

2468

2469 The qualifier is propagated on a CONTEXT message.

2470

2471 The “Qualifier name” shall be “transaction-timelimit”.

2472

2473 The “Content” shall contain the following field:

2474

Content field	Type
Timelimit	Integer

2475

2476 **Timelimit** indicates the maximum (further) duration, expressed as whole seconds from the
2477 time of transmission of the containing CONTEXT, of the active phase of the business
2478 transaction.

2479

2480 **Inferior timeout**

2481

2482 This qualifier allows an Inferior to limit the duration of its “promise”, when sending
2483 PREPARED, that it will maintain the ability to confirm or cancel the effects of all associated
2484 operations. Without this qualifier, an Inferior is expected to retain the ability to confirm or
2485 cancel indefinitely. If the timeout does expire, the Inferior is released from its promise and
2486 can apply the decision indicated in the qualifier.

2487

2488 It should be noted that BTP recognises the possibility that an Inferior may be forced to apply
2489 a confirm or cancel decision before the CONFIRM or CANCEL is received and before this

2490 timeout expires (or if this qualifier is not used). Such a decision is termed a heuristic decision,
2491 and (as with other transaction mechanisms), is considered to be an exceptional event. As with
2492 heuristic decisions, the taking of an autonomous decision by a Inferior **subsequent** to the
2493 expiry of this timeout, is liable to cause contradictory decisions across the business
2494 transaction. BTP ensures that at least the occurrence of such a contradiction will be
2495 (eventually) reported to the Superior of the business transaction. BTP treats “true” heuristic
2496 decisions and autonomous decisions after timeout the same way – in fact, the expiry in this
2497 timeout does not cause a qualitative (state table) change in what can happen, but rather a step
2498 change in the probability that it will.
2499

2500 The expiry of the timeout does not strictly require that the Inferior immediately invokes the
2501 intended decision, only that is at liberty to do so. An implementation may choose to only
2502 apply the decision if there is contention for the underlying resource, for example.
2503 Nevertheless, Superiors are recommended to avoid relying on this and ensure decisions for
2504 the business transaction are made before these timeouts expire (and allow a margin of error
2505 for network latency etc.).
2506

2507 The qualifier may be present on a PREPARED message. If the PREPARED message has the
2508 “default is cancel” parameter “true”, then the “IntendedDecision” field of this qualifier shall
2509 have the value “cancel”.
2510

2511 The “Qualifier name” shall be “inferior-timeout” .
2512

2513 The “Content” shall contain the following fields:
2514

Content field	Type
Timeout	Integer
IntendedDecision	“confirm” or “cancel”

2515
2516 **Timeout** indicates how long, expressed as whole seconds from the time of transmission of the
2517 carrying message, the Inferior intends to maintain its ability to either confirm or cancel the
2518 effects of the associated operations, as ordered by the receiving Superior.
2519

2520 **IntendedDecision** indicates which outcome will be applied, if the timeout completes and an
2521 autonomous decision is made.
2522

2523 **Minimum inferior timeout**

2524

2525 This qualifier allows a Superior to constrain the Inferior timeout qualifier received from the
2526 Inferior. If a Superior knows that the decision for the business transaction will not be
2527 determined for some period, it can require that Inferiors do not send PREPARED messages
2528 with Inferior timeouts that would expire before then. An Inferior that is unable or unwilling to
2529 send a PREPARED message with a longer (or no) timeout **should** cancel, and reply with
2530 CANCELLED.
2531

2532 The qualifier may be present on a CONTEXT, ENROLLED or PREPARE message. If
 2533 present on more than one, and with different values of the MinimumTimeout field, the value
 2534 on ENROLLED shall prevail over that on CONTEXT and the value on PREPARE shall
 2535 prevail over either of the others.

2536
 2537 The “Qualifier name” shall be “minimum-inferior-timeout” .

2538
 2539 The “Content” shall contain the following field:

Content field	Type
MinimumTimeout	Integer

2541
 2542 **Minimum Timeout** is the minimum value of timeout, expressed as whole seconds, that will be
 2543 acceptable in the Inferior timeout qualifier on an answering PREPARED message.

2544
 2545 **Inferior name**

2546
 2547 This qualifier allows an Enroller to supply a name for the Inferior that will be visible on
 2548 INFERIOR_STATUSES and thus allow the Terminator to determine which Inferior (of the
 2549 Composer or Coordinator) is related to which application work. This is in addition to the
 2550 “inferior handle” field. The name can be human-readable and can also be used in fault
 2551 tracing, debugging and auditing.

2552
 2553 The name is never used by the BTP actors themselves to identify each other or to direct
 2554 messages. (The BTP actors use the addresses and the identifiers in the message parameters
 2555 for those purposes.)

2556
 2557 This specification makes no requirement that the names are unambiguous within any scope
 2558 (unlike the “inferior-handle” on ENROLLED and BEGUN, which is required to be
 2559 unambiguous within the scope of the Decider). Other specifications, including those defining
 2560 use of BTP with a particular application may place requirements on the use and form of the
 2561 names. (This may include reference to information passed in application messages or in other,
 2562 non-standardised, qualifiers.)

2563
 2564 The qualifier may be present on BEGIN, ENROL and in the “qualifiers” field of a Status-item
 2565 in INFERIOR_STATUSES. It is present on BEGIN only if there is a related CONTEXT; if
 2566 present, the same qualifier value **should** be included in the consequent ENROL. If
 2567 INFERIOR_STATUSES includes a Status-item for an Inferior whose ENROL had an
 2568 inferior-name qualifier, the same qualifier value **should** be included in the Status-item.

2569
 2570 The “Qualifier -name” shall be “inferior-name”

2571
 2572 The “Content” shall contain the following fields:

Content field	Type
---------------	------

inferior-name String

2574

2575

2576

Inferior name the name assigned to the enrolling Inferior.

2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623

State Tables

Explanation of the state tables

The state tables deal with the state transitions of the Superior and Inferior roles and which message can be sent and received in each state. The state tables directly cover only a single, bi-lateral Superior:Inferior relationship. The interactions between, for example, multiple Inferiors of a single Superior that will apply the same decision to all or some (of them), are dealt with in the definitions of the “decision” events which also specify when changes are made to persistent state information (see below).

There are two state tables, one for Superior, one for Inferior. States are identified by a letter-digit pair, with upper-case letters for the superior, lower-case for the inferior. The same letter is used to group states which have the same, or similar, persistent state, with the digit indicating volatile state changes or minor variations. Corresponding upper and lower-case letters are used to identify (approximately) corresponding Superior and Inferior states.

The Inferior table includes events occurring both at the Inferior as such and at the associated Enroller, as the Enroller’s actions are constrained by and constrain the Inferior role itself.

Status queries

In BTP the messages SUPERIOR_STATE and INFERIOR_STATE are available to prompt the peer to report its current state by repeating the previous message (when this is allowed) or by sending the other *_STATE message. The “reply_requested” parameter of these messages distinguishes between their use as a prompt and as a reply. An implementation receiving a *_STATE message with “reply_requested” as “true” is not required to reply immediately – it may choose to delay any reply until a decision event occurs and then send the appropriate new message (e.g. on receiving INFERIOR_STATE/prepared/y while in state E1, a superior is permitted to delay until it has performed “decide to confirm” or “decide to cancel”). However, this may cause the other side to repeatedly send interrogatory *_STATE messages.

Note that a Superior (or some entity standing in for a now-extinct Superior) uses SUPERIOR_STATE/unknown to reply to messages received from an Inferior where the Superior:Inferior relationship is in an unknown (using state “Y1”). The *_STATE messages with a “state” value “inaccessible” can be used as a reply when **any** message is received and the implementation is temporarily unable to determine whether the relationship is known or what the state is. Other than these cases, the *_STATE messages with “reply requested” equal to “false” are only sent when the other message with “reply requested” equal to “true” has been received and no other message has been sent.

Decision events

The persistent state changes (equivalent to logging in a regular transaction system) and some other events are modelled as “decision events” (e.g. “decide to confirm”, “decide to be prepared”). The exact nature of the real events and changes in an implementation that are modelled by these events depends on the position of the Superior or Inferior within the business transaction and on features of the implementation (e.g. making of a persistent record

2624 of the decision means that the information will survive at least some failures that otherwise
2625 lose state information, but the level of survival depends on the purpose of the
2626 implementation). [Table 2](#) and [Table 3](#) define the decision events.

2627
2628 In some cases, an implementation may not need to make an active change to have a persistent
2629 record of a decision, provided that the implementation will restore itself to the appropriate
2630 state on recovery. For example, an (inferior) implementation that “decided to be prepared”,
2631 and recorded a timeout (to cancel) in the persistent information for that decision (signalled via
2632 the appropriate qualifier on PREPARED), could treat the presence of an expired record as a
2633 record of “decide to cancel autonomously”, provided it always updated such a record as part
2634 of the “apply ordered confirmation” decision event.

2635
2636 The Superior event “decide to prepare” is considered semi-persistent. Since the sending of
2637 PREPARE indicates that the application exchange (to associate operations with the Inferior)
2638 is complete, it is not meaningful for the Superior:Inferior relationship to revert to an earlier
2639 state corresponding to an incomplete application exchange. However, implementations are
2640 not required to make the sending of PREPARE persistent in terms of recovery – a Superior
2641 that experiences failure after sending PREPARE may, on recovery, have no information
2642 about the transaction, in which case it is considered to be in the completed state (Z), which
2643 will imply the cancellation of the Inferior and its associated operations.

2644
2645 Where a Superior is itself an Inferior (to another Superior entity), in a hierarchic tree, its
2646 “decide to confirm” and “decide to cancel” decisions will in fact be the receipt of a
2647 CONFIRM or CANCEL instruction from its own Superior, without necessary change of local
2648 persistent information (which would combine both superior and inferior information, pointing
2649 both up and down the tree).

2650
2651

2652 **Disruptions – failure events**

2653
2654 Failure events are modelled as “disruption”. A failure and the subsequent recovery will (or
2655 may) cause a change of state. The disruption events in the state tables model different extents
2656 of loss of state information. An implementation is not required to exhibit all the possible
2657 disruption events, but it is not allowed to exhibit state transitions that do not correspond to a
2658 possible disruption.

2659
2660 In addition to the disruption events in the tables, there is an implicit “disruption 0” event,
2661 which involves possible interruption of service and loss of messages in transit, but no change
2662 of state (either because no state information was lost, or because recovery from persistent
2663 information restores the implementation to the same state). The “disruption 0” event would
2664 typically be an appropriate abstraction for a communication failure.

2665

2666 **Invalid cells and assumptions of the communication mechanism**

2667
2668 The empty cells in state table represent events that cannot happen. For events corresponding
2669 to sending a message or any of the decision events, this prohibition is absolute – e.g. a
2670 conformant implementation in the Superior active state “B1” will not send CONFIRM. For

2671 events corresponding to receiving a message, the interpretation depends on the properties of
2672 the underlying communications mechanism.

2673

2674 For all communication mechanisms, it is assumed that

- 2675 a) the two directions of the Superior:Inferior communication are not synchronised –
- 2676 that is messages travelling in opposite directions can cross each other to any
- 2677 degree; any number of messages may be in transit in either direction; and
- 2678 b) messages may be lost arbitrarily

2679

2680 If the communication mechanisms guarantee ordered delivery (i.e. that messages, if delivered
2681 at all, are delivered to the receiver in the order they were sent) , then receipt of a message in a
2682 state where the corresponding cell is empty indicates that the far-side has sent a message out
2683 of order – a FAULT message with the Fault Type “WrongState” can be returned.

2684

2685 If the communication mechanisms cannot guarantee ordered delivery, then messages received
2686 where the corresponding cell is empty should be ignored. Assuming the far-side is
2687 conformant, these messages can assumed to be “stale” and have been overtaken by messages
2688 sent later but already delivered. (If the far-side is non-conformant, there is a problem
2689 anyway).

2690

2691 **Meaning of state table events**

2692

2693 The tables in this section define the events (rows) in the state tables. [Table 1](#) ~~Table 1~~ defines
2694 the events corresponding to sending or receiving BTP messages and the disruption events.
2695 [Table 2](#) ~~Table 2~~ describes the decision events for an Inferior, [Table 3](#) ~~Table 3~~ those for a
2696 Superior.

2697

2698 The decision events for a Superior, defined in [Table 3](#) ~~Table 3~~ cannot be specified without
2699 reference to other Inferiors to which it is Superior and to its relation with the application or
2700 other entity that (acting ultimately on behalf of the application) drives it.

2701

2702 The term “remaining Inferiors” refers to any actors to which this endpoint is Superior and
2703 which are to be treated as an atomic decision unit with (and thus including) the Inferior on
2704 this relationship. If the CONTEXT for this Superior:Inferior relationship had a “superior
2705 type” of “atom”, this will be all Inferiors established with same Superior address and Superior
2706 identifier except those from which RESIGN has been received. If the CONTEXT had
2707 “superior type” of “cohesion”, the “remaining Inferiors” excludes any that it has been
2708 determined will be cancelled, as well as any that have resigned – in other words it includes
2709 only those for which a confirm decision is still possible or has been made. The determination
2710 of exactly which Inferiors are “remaining Inferiors” in a cohesion is determined, in some
2711 way, by the application. The term “Other remaining Inferiors” excludes this Inferior on this
2712 relationship. A Superior with a single Inferior will have no “other remaining Inferiors”.

2713

2714 In order to ensure that the confirmation decision **is** delivered to all remaining Inferiors,
2715 despite failures, the Superior must persistently record which these Inferiors are (i.e. their
2716 addresses and identifiers). It must also either record that the decision is confirm, or ensure
2717 that the confirm decision (if there is one) is persistently recorded somewhere else, and that it

2718 will be told about it. This latter would apply if the Superior were also BTP Inferior to another
 2719 entity which persisted a confirm decision (or recursively deferred it still higher). However,
 2720 since there is no requirement that the Superior be also a BTP Inferior to any other entity, the
 2721 behaviour of asking another entity to make (and persist) the confirm decision is termed
 2722 "offering confirmation" - the Superior offers the possible confirmation of itself, and its
 2723 remaining Inferiors to some other entity. If that entity (or something higher up) then does
 2724 make and persist a confirm decision, the Superior is "instructed to confirm" (which is
 2725 equivalent BTP CONFIRM).

2726
 2727 The application, or an entity acting indirectly on behalf of the application, may request a
 2728 Superior to prepare an Inferior (or all Inferiors). This typically implies that there will be no
 2729 more operations associated with the Inferior. Following a request to prepare all remaining
 2730 Inferiors, the Superior may offer confirmation to the entity that requested the prepare. (If the
 2731 Superior is also a BTP Inferior, its superior can be considered an entity acting on behalf of the
 2732 application.)

2733
 2734 The application, or an entity acting indirectly on behalf of the application, may also request
 2735 confirmation. This means the Superior is to attempt to make and persist a confirm decision
 2736 itself, rather than offer confirmation.

2737
 2738
 2739

Table 1 : send, receive and disruption events

Event name	Meaning
send/receive ENROL/rsp-req	send/receive ENROL with reply-requested = true
send/receive ENROL/no-rsp-req	send/receive ENROL with reply-requested = false
send/receive RESIGN/rsp-req	send/receive RESIGN with reply-requested = true
send/receive RESIGN/no-rsp-req	send/receive RESIGN with reply-requested = false
send/receive PREPARED	send/receive PREPARED, with default-cancel = false
send/receive PREPARED/cancel	send/receive PREPARED, with default-cancel = true
send/receive CONFIRMED/auto	send/receive CONFIRMED, with confirm -received = true
send/receive CONFIRMED/response	send/receive CONFIRMED, with confirm -received = false
send/receive HAZARD	send/receive HAZARD
send/receive INF_STATE/***/y	send/receive INFERIOR_STATE with status *** and reply-requested = true
send/receive INF_STATE/***	send/receive INFERIOR_STATE with status *** and reply-requested = false

Event name	Meaning
send/receive SUP_STATE/***/y	send/receive SUPERIOR_STATE with status *** and reply-requested = true ("prepared-rcvd" represents "prepared-received")
send/receive SUP_STATE/***	send/receive SUPERIOR_STATE with status *** and reply-requested = false ("prepared-rcvd" represents "prepared-received")
disruption ***	Loss of state– new state is state applying after any local recovery processes complete

2740

2741

Table 2 : Decision events for Inferior

Event name	Meaning
decide to resign	<ul style="list-style-type: none"> Any associated operations have had no effect (data state is unchanged)).
decide to be prepared	<ul style="list-style-type: none"> Effects of all associated operations can be confirmed or cancelled; information to retain confirm/cancel ability has been made persistent
decide to be prepared/cancel	<ul style="list-style-type: none"> As "decide to be prepared"; the persistent information specifies that the default action will be to cancel
decide to confirm autonomously	<ul style="list-style-type: none"> Decision to confirm autonomously has been made persistent; the effects of associated operations will be confirmed regardless of failures
decide to cancel autonomously	<ul style="list-style-type: none"> Decision to cancel autonomously has been made persistent the effects of associated operations will be cancelled regardless of failures
apply ordered confirmation	<ul style="list-style-type: none"> Effects of all associated operations have been confirmed; Persistent information is effectively removed
remove persistent information	<ul style="list-style-type: none"> Persistent information is effectively removed;

Event name	Meaning
detect problem	<ul style="list-style-type: none"> ● For at least some of the associated operations, EITHER <ul style="list-style-type: none"> ○ they cannot be consistently cancelled or consistently confirmed; OR ○ it cannot be determined whether they will be cancelled or confirmed ● AND, information about this is not persistent
detect and record problem	<ul style="list-style-type: none"> ● As for the first condition of "detect problem" ● information recording this has been persisted (to the degree considered appropriate), or the detection itself is persistent. (i.e. will be re-detected on recovery)

2742

2743

Table 3: Decision events for a Superior

Event name	Meaning
decide to request confirm	<ul style="list-style-type: none"> ● All associated application messages to be sent to the service have been sent; ● There are no other remaining Inferiors ● All enrolments that would create other Inferiors have completed (no outstanding CONTEXT_REPLYS) ● The Superior has been requested to confirm
decide to prepare	<ul style="list-style-type: none"> ● All associated application messages to be sent to the service have been sent; ● The Superior has been requested to prepare this Inferior
decide to confirm	<ul style="list-style-type: none"> ● Either <ul style="list-style-type: none"> ○ PREPARED or PREPARED/cancel has been received from all other remaining Inferiors; AND ○ Superior has been requested to confirm; AND ○ persistent information records the confirm decision and identifies all remaining Inferiors; ● Or <ul style="list-style-type: none"> ○ persistent information records an offer of confirmation and has been instructed to confirm
decide to cancel	<ul style="list-style-type: none"> ● Superior has not offered confirmation; OR ● Superior has offered confirmation and has been instructed to cancel; OR ● Superior has offered confirmation but has made an

Event name	Meaning
	autonomous cancellation decision
remove confirm information record contradiction	<ul style="list-style-type: none"> • Persistent information has been effectively removed; • Information recording the contradiction has been persisted (to the degree considered appropriate)

2744

2745

2746

2747

2748

2749

2750

2751

2752

2753

2754

2755

2756

2757

2758

2759

2760

2761

2762

2763

2764

2765

2766

2767

2768

2769

2770

2771

2772

Persistent information

Persisted information (especially prepared information at an Inferior, confirm information at a Superior) may include qualifications of the state carried in Qualifiers of the corresponding message (e.g. inferior timeouts in prepared information). It may also include application-specific information (especially in Inferiors) to allow the future confirmation or cancellation of the associated operations. In some cases it will also include information allowing an application message sent with a BTP message (e.g. PREPARED) to be repeated.

The “effective” removal of persistent information allows for the possibility that the information is retained (perhaps for audit and tracing purposes) but some change to the persistent information (as a whole) means that if there is a failure after such change, on recovery, the persistent information does not cause the endpoint to return the state it would have recovered to before the change.

In all cases, the degree to which information described as “persistent” will survive failure is a configuration and implementation option. An implementation **should** describe the level of failure that it is capable of surviving. For applications manipulating information that is itself volatile (e.g. network configurations), there is no requirement to make the BTP state information more persistent than the application information.

The degree of persistence of the recording of a hazard (problem) at an Inferior and recording of a detected contradiction at a Superior may be different from that applying to the persistent prepared and confirm information. Implementations and configuration may choose to pass hazard and contradiction information via management mechanisms rather than through BTP. Such passing of information to a management mechanism could be treated as “record problem” or “record contradiction”.

Table 4 : Superior states

State	summary
I1	CONTEXT created
A1	ENROLing
B1	ENROLLED (active)
C1	resigning
D1	PREPARE sent
E1	PREPARED received
E2	PREPARED/cancel received
F1	CONFIRM sent
F2	completed after confirm
G1	cancel decided
G2	CANCEL sent
G3	cancelling, RESIGN received
G4	both cancelled
H1	inferior autonomously confirmed
J1	Inferior autonomously cancelled
K1	confirmed, contradiction detected
L1	cancelled, contradiction detected
P1	hazard reported
P2	hazard reported in null state
P3	hazard reported after confirm decision
P4	hazard reported after cancel decision
Q1	contradiction detected in null state
R1	Contradiction or hazard recorded
R2	completed after contradiction or hazard recorded
S1	REQUEST CONFIRM decided
Y1	completed queried
Z	completed and unknown

Table 5 : Inferior states

State	summary
i1	aware of CONTEXT
a1	enrolling
b1	enrolled
c1	resigning
d1	preparing
e1	prepared
e2	prepared,default to cancel
f1	confirming
f2	confirming after default cancel
g1	CANCEL received in prepared state
g2	CANCEL received in prepared/cancel state
h1	Autonomously confirmed
h2	autonomously confirmed, superior confirmed
j1	autonomously cancelled
j2	autonomously cancelled, superior cancelled
k1	autonomously cancelled, contradicted
k2	autonomously cancelled, CONTRADICTION received
l1	autonomously confirmed, contradicted
l2	autonomously confirmed, CONTRADICTION received
m1	confirm ation applied
n1	cancelling
p1	hazard detected, not recorded
p2	hazard detected in prepared state, not recorded
q1	hazard recorded
s1	REQUEST CONFIRM received after prepared state
s2	REQUEST CONFIRM received
s3	REQUEST CONFIRM received, confirming
s4	REQUEST CONFIRM received, cancelling
s5	REQUEST CONFIRM received, hazard detected
s6	REQUEST CONFIRM received, hazard recorded
x1	completed, presuming abort
x2	completed, presuming abort after prepared/cancel

State	summary
y1	completed, queried
y2	completed, default cancel, a message received
z	completed
z1	completed with default cancel

2776
2777

Table 6: Superior state table – normal forward progression

	I 1	A1	B1	C1	D1	E1	E2	F1	F2
receive ENROL/rsp-req	A1								
receive ENROL/no-rsp-req	B1								
receive RESIGN/rsp-req	Y1		C1	C1	C1				
receive RESIGN/no-rsp-req	Z		Z	Z	Z				
receive PREPARED	Y1		E1		E1	E1		F1	
receive PREPARED/cancel	Y1		E2		E2		E2	F1	
receive CONFIRMED/auto	Q1		H1		H1	H1		F1	
receive CONFIRMED/response								F2	F2
receive CANCELLED	Y1		Z		Z	J1	J1	K1	
receive HAZARD	P1	P1	P1		P1	P1	P1	P3	
receive INF_STATE/active/y	Y1	A1	B1		D1				
receive INF_STATE/active			B1		D1				
receive INF_STATE/unknown			Z	Z	Z				
send ENROLLED		B1							
send RESIGNED				Z					
send PREPARE					D1	E1	E2		
send CONFIRM_ONE_PHASE								F1	
send CONFIRM									
send CANCEL									
send CONTRADICTION									
send SUP_STATE/active/y			B1						
send SUP_STATE/active			B1						
send SUP_STATE/prepared-rcvd/y						E1	E2		
send SUP_STATE/prepared-rcvd						E1	E2		
send SUP_STATE/unknown									
decide to request confirm			S1			S1	S1		
decide to prepare			D1						
decide to confirm						F1	F1		
decide to cancel			G1		G1	G1	Z		
remove persistent information									Z
record contradiction									
disruption I	Z	Z	Z	Z	Z	Z	Z		F1
disruption II						D1	D1		
disruption III						B1	B1		
disruption IV									

Table 7: Superior state table – cancellation and contradiction

	G1	G2	G3	G4	H1	J1	K1	L1
receive ENROL/rsp-req								
receive ENROL/no-rsp-req								
receive RESIGN/rsp-req	G3	Z	G3					
receive RESIGN/no-rsp-req	Z	Z	Z					
receive PREPARED	G1	G2						
receive PREPARED/cancel	G1	G2						
receive CONFIRMED/auto	L1	L1			H1			L1
receive CONFIRMED/response								
receive CANCELLED	G4	Z		G4		J1	K1	
receive HAZARD	P4	P4						
receive INF_STATE/active/y	G1	G2						
receive INF_STATE/active	G1	G2						
receive INF_STATE/unknown	Z	Z	Z	Z				
send ENROLLED								
send RESIGNED								
send PREPARE								
send CONFIRM_ONE_PHASE								
send CONFIRM								
send CANCEL	G2	G2	Z	Z				
send CONTRADICTION								
send SUP_STATE/active/y								
send SUP_STATE/active								
send SUP_STATE/prepared-rcvd/y								
send SUP_STATE/prepared-rcvd								
send SUP_STATE/unknown								
decide to request confirm								
decide to prepare								
decide to confirm					F1	K1		
decide to cancel					L1	G4		
remove persistent information								
record contradiction							R1	R1
disruption I	Z	Z	Z	Z	Z	Z	F1	Z
disruption II			G2	G2	E1	E1		G2
disruption III					D1	D1		
disruption IV					B1	B1		

Table 8: Superior state table – hazard and request confirm

	P1	P2	P3	P4	Q1	R1	R2	S1
receive ENROL/rsp-req								
receive ENROL/no-rsp-req								
receive RESIGN/rsp-req								C1
receive RESIGN/no-rsp-req								Z
receive PREPARED								S1
receive PREPARED/cancel								S1
receive CONFIRMED/auto					Q1	R1	R1	S1
receive CONFIRMED/response					Z	R2		Z
receive CANCELLED						R1	R1	Z
receive HAZARD	P1	P2	P3	P4		R1	R1	Z
receive INF_STATE/active/y								S1
receive INF_STATE/active								S1
receive INF_STATE/unknown	P1	P2		P4		R2	R2	Z
send ENROLLED								
send RESIGNED								
send PREPARE								
send CONFIRM_ONE_PHASE								S1
send CONFIRM								
send CANCEL								
send CONTRADICTION						R2		
send SUP_STATE/active/y								
send SUP_STATE/active								
send SUP_STATE/prepared-rcvd/y								
send SUP_STATE/prepared-rcvd								
send SUP_STATE/unknown								
decide to request confirm								
decide to prepare								
decide to confirm								
decide to cancel								
remove persistent information							Z	
record contradiction	R1	R1	R1	R1	R1			
disruption I	Z	Z	Z	Z	Z		R1	Z
disruption II	D1		F1	G2				
disruption III	B1							
disruption IV								

Table 9: Superior state table – query after completion and completed states

	Y1	Z
receive ENROL/rsp-req		Y1
receive ENROL/no-rsp-req		Y1
receive RESIGN/rsp-req	Y1	Y1
receive RESIGN/no-rsp-req	Z	Z
receive PREPARED	Y1	Y1
receive PREPARED/cancel	Y1	Y1
receive CONFIRMED/auto	Q1	Q1
receive CONFIRMED/response	Z	Z
receive CANCELLED	Y1	Y1
receive HAZARD	P2	P2
receive INF_STATE/active/y	Y1	Y1
receive INF_STATE/active	Y1	Z
receive INF_STATE/unknown	Z	Z
send ENROLLED		
send RESIGNED		
send PREPARE		
send CONFIRM_ONE_PHASE		
send CONFIRM		
send CANCEL		
send CONTRADICTION		
send SUP_STATE/active/y		
send SUP_STATE/active		
send SUP_STATE/prepared-rcvd/y		
send SUP_STATE/prepared-rcvd		
send SUP_STATE/unknown	Z	
decide to request confirm		
decide to prepare		
decide to confirm		
decide to cancel		
remove persistent information		
record contradiction		
disruption I	Z	
disruption II		
disruption III		
disruption IV		

2781
2782

2782

2783

Table 10: Inferior state table– normal forward progression

	i 1	a1	b1	c1	d1	e1	e2	f1	f2
send ENROL/rsp-req	a1								
send ENROL/no-rsp-req	b1								
send RESIGN/rsp-req				c1					
send RESIGN/no-rsp-req				z					
send PREPARED						e1			
send PREPARED/cancel							e2		
send CONFIRMED/auto									
send CONFIRMED/response									
send CANCELLED			z		z				
send HAZARD									
send INF_STATE/active/y		a1	b1		d1				
send INF_STATE/active			b1		d1				
send INF_STATE/unknown									
receive ENROLLED		b1							
receive RESIGNED				z					
receive PREPARE		d1	d1	c1	d1	e1	e2		
receive CONFIRM_ONE_PHASE		s2	s2	c1		s1	s1		
receive CONFIRM						f1	f2	f1	f2
receive CANCEL		n1	n1	z	n1	g1	g2		
receive CONTRADICTION									
receive SUP_STATE/active/y		b1	b1	c1		e1	e2		
receive SUP_STATE/active		b1	b1	c1		e1	e2		
receive SUP_STATE/prepared-rcvd/y						e1	e2		
receive SUP_STATE/prepared-rcvd						e1	e2		
receive SUP_STATE/unknown		z	z	z	z	x1	x2		
decide to resign				c1	c1				
decide to be prepared				e1	e1				
decide to be prepared/cancel				e2	e2				
decide to confirm autonomously						h1			
decide to cancel autonomously						j1	z1		
apply ordered confirmation								m1	m1
remove persistent information									
detect problem		p1	p1		p1	p2	p2	p2	p2
detect and record problem									
disruption I		z	z	z	z			e1	e2
disruption II					b1				
disruption III									

2784

2785

Table 11: Inferior state table– cancellation and contradiction

	g1	g2	h1	h2	j1	j2	k1	k2	l1	l2
send ENROL/rsp-req send ENROL/no-rsp-req send RESIGN/rsp-req send RESIGN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD			h1		j1		k1		l1	
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown										
receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION			h1 s3 h2	h2	j1 s4 k1	j2	k1		l1	
receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown			h1 h1 h1 h1		j1 j1 j1 j1	j2		k2	l1	
decide to resign decide to be prepared decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect problem detect and record problem				m1		z		z		z
disruption I disruption II disruption III	e1	e2	h1		j1		j1 k1 j1		h1	l1 h1

2786
2787

Table 12: Inferior state table– confirm, cancel ordered and hazard recording

	m1	n1	p1	p2	q1
send ENROL/rsp-req send ENROL/no-rsp-req send RESIGN/rsp-req send RESIGN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD	z	z	p1	p2	q1
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown					
receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION	m1	n1	p1 p1 s5	q1 p2 s5	q1 q1 s6
receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown		z	p1 p1	p2 p2	q1 q1 q1 q1
decide to resign decide to be prepared decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect problem detect and record problem					q1 q1
disruption I disruption II disruption III	z	z d1 b1	z		

Table 13: Inferior state table– request confirm states

	s1	s2	s3	s4	s5	s6
send ENROL/rsp-req send ENROL/no-rsp-req send RESIGN/rsp-req send RESIGN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD			z	z	z	z
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown						
receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION	s1	s2	s3	s4	s5	s6
receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown	x1	z	z	z	z	z
decide to resign decide to be prepared decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect problem detect and record problem			s3 s4			s2 s6
disruption I disruption II disruption III	e1	z		z	z	

2790

Table 14: Inferior state table– completed states (including presume -abort and queried)

	x1	x2	y1	y2	z	z1
send ENROL/rsp-req send ENROL/no-rsp-req send RESIGN/rsp-req send RESIGN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD						z1
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown					z	
receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION					z	
receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown			y1	y2	y1	y2
			y1	y2	z	z1
				y2		y2
				y2		y2
	x1	x2	y1	y2	z	z
decide to resign decide to be prepared decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect problem detect and record problem						
	z	z				
disruption I disruption II disruption III	e1	e2				

2791

2792

2793

2793 **Failure Recovery**

2794 **Types of failure**

2795

2796 BTP is designed to ensure the delivery of a consistent decision for a business transaction to
2797 the parties involved, even in the event of failure. Failures can be classified as:

2798

2799

2800

2801

2802

2803

2804

2805

2806

2807

2808

Communication failure: messages between BTP actors are lost and not delivered. BTP assumes the carrier protocol ensures that messages are either delivered correctly (without corruption) or are lost, but does not assume that all losses are reported or that messages sent separately are delivered in the order of sending.

2809

2810

2811

2812

2813

2814

2815

2816

2817

2818

2819

2820

Communication failure may become known to a BTP implementation by an indication from the lower layers or may be inferred (or suspected) by the expiry of a timeout. Recovery from a communication failure requires only that the two actors can again send messages to each other and continue or complete the progress of the business transaction. In the state tables for the Superior:Inferior relationship, each side is either waiting to make a decision or can send a message. For some states, the message to be sent is a repetition of a regular message; for other states, the INFERIOR_STATE or SUPERIOR_STATE message can be sent, requesting a response. Thus, following a communication failure, either side can prompt the other to re-establish the relationship. Receiving one of the *_STATE messages asking for a response does not require an immediate response – especially if an implementation is waiting to determine a decision (perhaps because it is itself waiting for a decision from elsewhere), an implementation may choose not to reply until it wishes too.

2821

2822

2823

2824

2825

2826

2827

2828

2829

2830

2831

A node failure is distinguished from communication failure because there is loss of volatile state. To ensure consistent application of the decision of a business transaction, BTP requires that some state information will be persisted despite node failure. Exactly what real events correspond to node failure but leave the persistent information undamaged is a matter for implementation choice, depending on application requirements; however, for most application uses, power failure should be survivable (an exception would be if the data manipulated by the associated operations was volatile). There will always be some level of event sufficiently catastrophic to lose persistent information and the ability to recover—destruction of the computer or bankruptcy of the organisation, for example.

2832

2833

2834

2835

2836

2837

2838

Recovery from node failure involves recreating the endpoint in a node that has access to the persistent information for incomplete transactions. This may be a recreation of the original node (including the ability to perform application work) using the same addresses; or there may be a distinct recovery entity, which can access the persistent data, but has a different address; other implementation approaches are possible. Restoration of the endpoint from persistent information will often result in a partial loss of state, relative to the volatile state reached before the failure. This is modelled in the state tables by the “disruption” events.

2839 After recovery from node failure, the implementation behaves much as if a communication
2840 failure had occurred.

2841

2842 Persistent information

2843

2844 BTP requires that some decision events are persisted – that information recording an
2845 Inferior’s decision to be prepared, a Superior’s decision to confirm and an Inferior’s
2846 autonomous decision survive failure. Making the first two decisions persistent ensures that a
2847 consistent decision can be reached for the business transaction and that it is delivered to all
2848 involved nodes. Requiring an Inferior’s autonomous decision to be persistent allows BTP to
2849 ensure that, if this decision is contradictory (i.e. opposite to the decision at the Superior), the
2850 contradiction will be reported to the Superior, despite failures.

2851

2852 BTP also permits, but does not require, recovery of the Superior:Inferior relationship in the
2853 active state (unlike many transaction protocols, where a communication or endpoint failure in
2854 active state would invariably cause rollback of the transaction). Recovery in the active state
2855 may require that the application exchange is resynchronised as well – BTP does not directly
2856 support this, but does allow continuation of the business transaction as such. In the state
2857 tables, from some states, there are several levels of disruption, distinguished by which state
2858 the implementation transits to – this represents the survival of different extents of state
2859 information over failure and recovery. The different levels of disruption describe legitimate
2860 states for the endpoint to be in after it has recovered – **they do not require that all
2861 implementations are able to exhibit the appropriate partial loss of state information.**

2862 The absence of a destination state for the disruption events means that such a transition is not
2863 legitimate – thus, for example, an Inferior that has decided to be prepared will always recover
2864 to the same state, by virtue of the information persisted in the “decide to be prepared” event.
2865

2866 Apart from the (optional) recovery in active state, BTP follows the well-known presume -
2867 abort model – it is only required that information be persisted when decisions are made (and
2868 not, e.g. on enrolment). This means that on recovery, one side may have persistent
2869 information but the other does not. This occurs when an Inferior has decided to be prepared
2870 but the Superior never confirmed (so the decision is “presumed” to be cancel), or because the
2871 Superior did confirm, and the Inferior applied the confirm, removed its persistent information
2872 but the acknowledgement (CONFIRMED) was never received by the Superior (or, at least, it
2873 still had the persistent information when the failure occurred).
2874

2875 Information to be persisted for an Inferior’s “decision to be prepared” must be sufficient to
2876 re-establish communication with the Superior, to apply a confirm decision and to apply a
2877 cancel decision. It will thus need to include

2878 Inferior identity (this may be an index used to locate the information)

2879 Superior address (as on CONTEXT)

2880 Superior identifier (as on CONTEXT)

2881 default -is-cancel value (as on PREPARED)

2882

2883 The information needed to apply confirm/cancel decisions will depend on the application and
2884 the associated operations. It may also normally be necessary to persist any qualifiers that

2885 were sent with the PREPARED message or application messages sent with the PREPARED,
2886 since the PREPARED message will be repeated if a failure occurs.

2887

2888 A Superior must record corresponding information to allow it to re-establish communication
2889 with the Inferior:

2890 Inferior address (as on ENROL)

2891 Inferior identifier (as on ENROL)

2892

2893 A Superior that is the Decider for the business transaction need only persist this information
2894 if it makes a decision to confirm (and this Inferior is in the confirm set, for a Cohesion). A
2895 Superior that is also an Inferior to some other entity (i.e. it is an intermediate in a tree, as
2896 atom in a cohesion, sub-coordinator or sub-composer) must persist this information as
2897 Superior (to this Inferior) as part of the persistent information of its decision to be prepared
2898 (as an Inferior). For such an entity, the “decision to confirm” as Superior is made when (and
2899 if) CONFIRM is received from its Superior or it makes an autonomous decision to confirm. If
2900 CONFIRM is received, the persistent information may be changed to show the confirm
2901 decision, but alternatively, the receipt of the CONFIRM can be treated as the decision itself.
2902 If the persistent information is left unchanged and there is a node failure, on recovery the
2903 entity (as an Inferior) will be in a prepared state, and will rediscover the confirm decision
2904 (using the recovery exchanges to its Superior) before propagating it to its Inferior(s).

2905

2906 After failure, an implementation may not be able to restore an endpoint to the appropriate
2907 state immediately – in particular, the necessary persistent information may be inaccessible,
2908 although the implementation can respond to received BTP messages. In such a case, a
2909 Superior may reply to any BTP message except INFERIOR_STATE/* (i.e. with a “reply-
2910 requested” value “false”) with SUPERIOR_STATE/inaccessible and an Inferior to any BTP
2911 message except SUPERIOR_STATE/* with “INFERIOR_STATE/inaccessible. Receipt of
2912 the *_STATE/inaccessible messages has no effect on the endpoint state.

2913

2914 Redirection

2915

2916 As described above, BTP uses the presume-abort model for recovery. A corollary of this is
2917 that there are cases where one side will attempt to re-establish communication when there is
2918 no persistent information for the relationship at the far-end. In such cases, it is important the
2919 side that is attempting recovery can distinguish between unsuccessful attempts to connect to
2920 the holder of the persistent information and when the information no longer exists. If the peer
2921 information does not exist, this side can draw conclusions and complete appropriately; if they
2922 merely fail to get through they are stuck in attempting recovery.

2923

2924 Two mechanisms are provided to make it possible that even when one side of a
2925 Superior:Inferior relationship has completed, that a message can eventually get through to
2926 something that can definitively report the status, distinguishing this case from a temporary
2927 inability to access the state of a continuing transaction element. The mechanisms are:

- 2928 o Address fields which provide a “callback address” can be a set of addresses,
2929 which are alternatives one of which is chosen as the target address for the
2930 future message. If the sender of that message finds the address does not work,
2931 it can try a different alternative.

2932 o The REDIRECT message can be used to inform the peer that an address
2933 previously given is no longer valid and to supply a replacement address (or
2934 set of addresses). REDIRECT can be issued either as a response to receipt of
2935 a message or spontaneously.

2936
2937 The two mechanisms can be used in combination, with one or more of the original set of
2938 addresses just being a redirector, which does not itself ever have direct access to the state
2939 information for the transaction, but will respond to any message with an appropriate
2940 REDIRECT.

2941
2942 An alternative implementation approach is to have a single addressable entity that uses the
2943 same address for all transactions, distinguishing them by identifier, and which always
2944 recovers to use the same address. Such an implementation would not need to supply
2945 “backup” addresses (and would only use REDIRECT if it was being permanently migrated).

2946

2947 Terminator:Decider failures

2948

2949 BTP does not provide facilities or impose requirements on the recovery of
2950 Terminator:Decider relationships, other than allowing messages to be repeated. A Terminator
2951 may survive failures (by retaining knowledge of the Decider’s address and identifier), but this
2952 is an implementation option. Although a Decider (if it decides to confirm) will persist
2953 information about the confirm decision, it is not required, after failure, to remain accessible
2954 using the inferior address it offered to the Terminator. Any such recovery is an
2955 implementation option.

2956

2957 A Decider’s address (as returned on BEGUN) may be a set of addresses, allowing a failed
2958 Decider to be recovered at a different address.

2959

2960 A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and
2961 thus no way of detecting that a Terminator has failed. To avoid a Decider waiting for ever for
2962 a REQUEST_CONFIRM that will never arrive, the standard qualifier “Transaction timelimit”
2963 can be used (by the Initiator) to inform the Decider when it can assume the Terminator will
2964 not issue REQUEST_CONFIRM and so it (the Decider) should initiate cancellation.

2965

2966 XML representation of Message Set

2967

2968 This section describes the syntax for BTP messages in XML. These XML messages represent
2969 a midpoint between the abstract messages and what actually gets sent on the wire.

2970

2971 All BTP related URIs have been created using Oasis URI conventions as specified in [RFC](#)
2972 [3121](#)

2973

2974 The XML Namespace for the BTP messages is urn:oasis:names:tc:BTP:xml

2975

2976 In addition to an XML schema, this specification uses an informal syntax to describe the
2977 structure of the BTP messages. The syntax appears as an XML instance, but the values
2978 contain data types instead of values. The following symbols are appended to some of the

2979 XML constructs: ? (zero or one), * (zero or more), + (one or more.) The absence of one of
2980 these symbols corresponds to "one and only one."

2981

2982 Addresses

2983

2984 As described in the “Abstract Message and Associated Contracts – Addresses” section, a BTP
2985 address comprises three parts, and for a target address only the “additional information” field
2986 is inside the BTP messages. For all BTP messages whose abstract form includes a target
2987 address parameter, the corresponding XML representation includes a “target-additional-
2988 information” element. This element may be omitted if it would be empty.

2989

2990 For other addresses, all three fields are present, as in:

2991

2992

```
<btp:some-address>  
  <btp:binding-name>...carrier binding URI...</btp:binding-name>  
  <btp:binding-address>...carrier specific  
  address...</btp:binding-address>  
  <btp:additional-information>...optional additional addressing  
  information...</btp:additional-information> ?  
</btp:some-address>
```

2993

2994

2995

2996

2997

2998

2999

3000

3001

3002

3003

3004

3005

3006

3007

3008

3009

A "published" address can be a set of <some-address>, which are alternatives which can be chosen by the peer (sender.) Multiple addresses are used in two cases: different bindings to same endpoint, or backup endpoints. In the former, the receiver of the message has the choice of which address to use (depending on which binding is preferable.) In the case where multiple addresses are used for redundancy, a **priority** attribute can be specified to help the receiver choose among the addresses- the address with the highest priority should be used, other things being equal. The **priority** is used as a hint and does not enforce any behaviour in the receiver of the message. Default priority is a value of 1.

3010

3010 Qualifiers

3011

3012

3013

The “Qualifier name” is used as the element name, within the namespace of the “Qualifier group”.

3014

3014 Examples:

3015

3016

3017

3018

3019

3020

3021

3022

3023

3024

3025

3026

```
<btpq:inferior-timeout  
  xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers"  
  xmlns:btp="urn:oasis:names:tc:BTP:xml "  
  btp:must-be-understood="false"  
  btp:to-be-propagated="false">1800</auth:username>  
  
<auth:username  
  xmlns:auth="http://www.example.com/ns/auth"  
  xmlns:btp="urn:oasis:names:tc:BTP:xml "  
  btp:must-be-understood="true"  
  btp:to-be-propagated="true">jtauber</auth:username>
```

3027

3028

Attributes **must-be-understood** has default value “true” and **to-be-propagated** has default value “false”.

3029

3030

Identifiers

3031

Unspecified length strings made of up hexadecimal digits (0->9, A->F). Note: lower case a->f are not valid.

3032

3033

3034

Examples: "01", "FAB224234CCCC2"

3035

3036

Note – Use of hexadecimal digits avoids problems with character-code representations. The only operation the BTP implementations have to perform on identifiers is to match them.

3037

3038

3039

Message References

3040

Each BTP message has an optional **id** attribute to give it a unique identifier. An application can make use of those identifiers, but no processing is enforced.

3041

3042

3043

Messages

3044

3045

CONTEXT

3046

3047

```
<btp:context id? superior-type="cohesion|atom">
  <btp:superior-address> +
  ...address...
</btp:superior-address>
<btp:superior-identifier>...hexstring...</btp:superior-
identifier>
<btp:qualifiers> ?
  ...qualifiers...
</btp:qualifiers>
</btp:context>
```

3048

3049

3050

3051

3052

3053

3054

3055

3056

3057

3058

3059

CONTEXT-REPLY

3060

3061

```
<btp:context-reply id? superior-type="cohesion|atom">
  <btp:superior-address> +
  ...address...
</btp:superior-address>
<btp:superior-identifier>...hexstring...</btp:superior-
identifier>
<completion-status>completed|related|repudiated</completion-
status>
<btp:qualifiers> ?
  ...qualifiers...
</btp:qualifiers>
</btp:context>
```

3062

3063

3064

3065

3066

3067

3068

3069

3070

3071

3072

3073

3074

3075

BEGIN

3076

3077

```
<btp:begin id? transaction-type="cohesion|atom">
  <btp:target-additional-information>
```

3078

```

3079     ...additional address information...
3080 </btp:target-additional-information>
3081 <btp:reply-address>
3082     ...address...
3083 </btp:reply-address>
3084 <btp:qualifiers> ?
3085     ...qualifiers...
3086 </btp:qualifiers>
3087 </btp:begin>
3088
3089

```

BEGUN

```

3090
3091
3092 <btp:begin id? transaction-type="cohesion|atom">
3093     <btp:target-additional-information>
3094         ...additional address information...
3095     </btp:target-additional-information>
3096     <btp:decider-address> ?
3097         ...address...
3098     </btp:decider-address>
3099     <btp:transaction-identifier>...hexstring...</btp:transaction-
3100 identifier> ?
3101     <btp:inferior-handle>...hexstring...</btp:inferior:handle> ?
3102     <btp:inferior-address> ?
3103         ...address...
3104     </btp:inferior-address>
3105     <btp:qualifiers> ?
3106         ...qualifiers...
3107     </btp:qualifiers>
3108 </btp:begin>
3109
3110

```

ENROL

```

3111
3112
3113 <btp:enrol reply-requested="true|false" id?>
3114     <btp:target-additional-information>
3115         ...additional address information...
3116     </btp:target-additional-information>
3117     <btp:superior-identifier>...hexstring...</btp:superior-
3118 identifier>
3119     <btp:reply-address> ?
3120         ...address...
3121     </btp:reply-address>
3122     <btp:inferior-address> +
3123         ...address...
3124     </btp:inferior-address>
3125     <btp:inferior-identifier>...hexstring...</btp:inferior-
3126 identifier>
3127     <btp:qualifiers> ?
3128         ...qualifiers...
3129     </btp:qualifiers>
3130 </btp:enrol>
3131

```

3132

3133

ENROLLED

3134

3135

```
<btp:enrolled id?>
```

3136

```
<btp:target-additional-information>
```

3137

```
  ..additional address information...
```

3138

```
</btp:target-additional-information>
```

3139

```
<btp:inferior-identifier>...hexstring...</btp:inferior-  
identifier>
```

3140

```
<btp:inferior-handle>...hexstring...</btp:inferior:handle> ?
```

3141

```
<btp:qualifiers> ?
```

3142

```
  ..qualifiers...
```

3143

```
</btp:qualifiers>
```

3144

```
</btp:enrolled>
```

3145

3146

3147

3148

RESIGN

3149

```
<btp:resign response-requested="true|false" id?>
```

3150

```
<btp:target-additional-information>
```

3151

```
  ..additional address information...
```

3152

```
</btp:target-additional-information>
```

3153

```
<btp:superior-identifier>...hexstring...</btp:superior-  
identifier>
```

3154

```
<btp:inferior-address> +
```

3155

```
  ..address...
```

3156

```
</btp:inferior-address>
```

3157

```
<btp:inferior-identifier>...hexstring...</btp:inferior-  
identifier>
```

3158

```
<btp:qualifiers> ?
```

3159

```
  ..qualifiers...
```

3160

```
</btp:qualifiers>
```

3161

```
</btp:resign>
```

3162

3163

3164

3165

3166

3167

RESIGNED

3168

```
<btp:resigned id?>
```

3169

```
<btp:target-additional-information>
```

3170

```
  ..additional address information...
```

3171

```
</btp:target-additional-information>
```

3172

```
<btp:inferior-identifier>...hexstring...</btp:inferior-  
identifier>
```

3173

```
<btp:qualifiers> ?
```

3174

```
  ..qualifiers...
```

3175

```
</btp:qualifiers>
```

3176

```
</btp:resigned>
```

3177

3178

3179

3180

3181

PREPARE

3182

```
<btp:prepare id?>
```

3183

```

3184 <btp:target-additional-information>
3185   ..additional address information...
3186 </btp:target-additional-information>
3187 <btp:inferior-identifier>...hexstring...</btp:inferior-
3188 identifier> ?
3189 <btp:reply-address> ?
3190   ..address...
3191 </btp:reply-address>
3192 <btp:transaction-identifier>...hexstring...</btp:transaction-
3193 identifier> ?
3194 <btp:inferiors-list> ?
3195   <btp:inferior-handle>...hexstring...</btp:inferior-handle>
3196 +
3197 </btp:inferiors-list>
3198 <btp:qualifiers> ?
3199   ..qualifiers...
3200 </btp:qualifiers>
3201 </btp:prepare>
3202
3203

```

PREPARED

```

3204
3205
3206 <btp:prepared default-is-cancel="false|true" id?>
3207 <btp:target-additional-information>
3208   ..additional address information...
3209 </btp:target-additional-information>
3210 <btp:superior-identifier>...hexstring...</btp:superior-
3211 identifier>
3212 <btp:inferior-address> +
3213   ..address...
3214 </btp:inferior-address>
3215 <btp:inferior-identifier>...hexstring...</btp:inferior-
3216 identifier>
3217 <btp:qualifiers> ?
3218   ..qualifiers...
3219 </btp:qualifiers>
3220 </btp:prepared>
3221
3222

```

CONFIRM

```

3223
3224
3225 <btp:confirm id?>
3226 <btp:target-additional-information>
3227   ..additional address information...
3228 </btp:target-additional-information>
3229 <btp:inferior-identifier>...hexstring...</btp:inferior-
3230 identifier>
3231 <btp:qualifiers> ?
3232   ..qualifiers...
3233 </btp:qualifiers>
3234 </btp:confirm>
3235

```

3236

3237

CONFIRMED

3238

3239

```
<btp:confirmed confirmed-received="true|false" id?>
```

3240

```
<btp:target-additional-information>
```

3241

```
..additional address information...
```

3242

```
</btp:target-additional-information>
```

3243

```
<btp:superior-identifier>...hexstring...</btp:superior-  
identifier>
```

3244

3245

```
<btp:inferior-address> ?
```

3246

```
..address...
```

3247

```
</btp:inferior-address>
```

3248

```
<btp:inferior-identifier>...hexstring...</btp:inferior-  
identifier> ?
```

3249

3250

```
<btp:decider-address> ?
```

3251

```
..address...
```

3252

```
</btp:decider-address>
```

3253

```
<btp:transaction-identifier>...hexstring...</btp:transaction-  
identifier> ?
```

3254

3255

```
<btp:qualifiers> ?
```

3256

```
..qualifiers...
```

3257

```
</btp:qualifiers>
```

3258

```
</btp:confirmed>
```

3259

3260

CANCEL

3261

3262

```
<btp:cancel id?>
```

3263

```
<btp:target-additional-information>
```

3264

```
..additional address information...
```

3265

```
</btp:target-additional-information>
```

3266

```
<btp:inferior-identifier>...hexstring...</btp:inferior-  
identifier> ?
```

3267

3268

```
<btp:reply-address> ?
```

3269

```
..address...
```

3270

```
</btp:reply-address>
```

3271

```
<btp:transaction-identifier>...hexstring...</btp:transaction-  
identifier> ?
```

3272

3273

```
<btp:inferiors-list> ?
```

3274

```
<btp:inferior-handle>...hexstring...</btp:inferior-handle>
```

3275

```
</btp:inferiors-list>
```

3276

```
<btp:qualifiers> ?
```

3277

```
..qualifiers...
```

3278

```
</btp:qualifiers>
```

3279

```
</btp:cancel>
```

3280

3281

3282

CANCELLED

3283

3284

```
<btp:cancelled id?>
```

3285

```
<btp:target-additional-information>
```

3286

```
..additional address information...
```

3287

```

3288     </btp:target-additional-information>
3289     <btp:superior-identifier>...hexstring...</btp:superior-
3290 identifier>
3291     <btp:inferior-address> +
3292     ...address...
3293     </btp:inferior-address> ?
3294     <btp:inferior-identifier>...hexstring...</btp:inferior-
3295 identifier> ?
3296     <btp:decider-address> ?
3297     ...address...
3298     </btp:decider-address>
3299     <btp:transaction-identifier>...hexstring...</btp:transaction-
3300 identifier> ?
3301     <btp:qualifiers> ?
3302     ...qualifiers...
3303     </btp:qualifiers>
3304 </btp:cancelled>
3305
3306

```

HAZARD

```

3307
3308
3309 <btp:hazard level="mixed|possible" id?>
3310   <btp:target-additional-information>
3311     ...additional address information...
3312   </btp:target-additional-information>
3313   <btp:superior-identifier>...hexstring...</btp:superior-
3314 identifier>
3315   <btp:inferior-address> +
3316   ...address...
3317   </btp:inferior-address>
3318   <btp:inferior-identifier>...hexstring...</btp:inferior-
3319 identifier>
3320   <btp:qualifiers> ?
3321   ...qualifiers...
3322   </btp:qualifiers>
3323 </btp:hazard>
3324
3325

```

CONTRADICTION

```

3326
3327
3328 <btp:contradiction id?>
3329   <btp:target-additional-information>
3330     ...additional address information...
3331   </btp:target-additional-information>
3332   <btp:inferior-identifier>...hexstring...</btp:inferior-
3333 identifier>
3334   <btp:qualifiers> ?
3335   ...qualifiers...
3336   </btp:qualifiers>
3337 </btp:contradiction>
3338
3339

```

SUPERIOR_STATE

3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356

```
<btp:superior-state reply-requested="true|false" id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:inferior-identifier>...hexstring...</btp:inferior-
identifier>
  <btp:status>active|prepared-
received|inaccessible|unknown</btp:status>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:superior-state>
```

INFERIOR_STATE

3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378

```
<btp:inferior-state reply-requested="true|false" id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:superior-identifier>...hexstring...</btp:superior-
identifier>
  <btp:inferior-address> +
    ...address...
  </btp:inferior-address>
  <btp:inferior-identifier>...hexstring...</btp:inferior-
identifier>
  <btp:status> active|prepared-
received|inaccessible|unknown</btp:status>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:inferior-state>
```

CONFIRM_ONE_PHASE

3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391

```
<btp:confirm-one-phase report-hazard="true|false" id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:inferior-identifier>...hexstring...</btp:inferior-
identifier>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:confirm-one-phase>
```

3392

3393

REQUEST_CONFIRM

3394

3395

```
<btp:request_confirm report-hazard="true|false" id?>
```

3396

```
<btp:target-additional-information>
```

3397

```
..additional address information...
```

3398

```
</btp:target-additional-information>
```

3399

```
<btp:reply-address>
```

3400

```
..address...
```

3401

```
</btp:reply-address>
```

3402

```
<btp:transaction-identifier>...hexstring...</btp:transaction-  
identifier>
```

3403

3404

```
<btp:inferiors-list> ?
```

3405

```
<btp:inferior-handle>...hexstring...</btp:inferior-handle>
```

3406

```
+
```

3407

```
</btp:inferiors-list>
```

3408

```
<btp:qualifiers> ?
```

3409

```
..qualifiers...
```

3410

```
</btp:qualifiers>
```

3411

```
</btp:request_confirm>
```

3412

3413

3414

REQUEST_STATUSES

3415

```
<btp:request_statuses id?>
```

3416

```
<btp:target-additional-information>
```

3417

```
..additional address information...
```

3418

```
</btp:target-additional-information>
```

3419

```
<btp:reply-address>
```

3420

```
..address...
```

3421

```
</btp:reply-address>
```

3422

```
<btp:transaction-identifier>...hexstring...</btp:transaction-  
identifier>
```

3423

3424

```
<btp:inferiors-list> ?
```

3425

```
<btp:inferior-handle>...hexstring...</btp:inferior-handle>
```

3426

```
+
```

3427

```
</btp:inferiors-list>
```

3428

```
<btp:qualifiers> ?
```

3429

```
..qualifiers...
```

3430

```
</btp:qualifiers>
```

3431

```
</btp:request_statuses>
```

3432

3433

3434

3435

INFERIOR_STATUSES

3436

```
<btp:inferior_statuses id?>
```

3437

```
<btp:target-additional-information>
```

3438

```
..additional address information...
```

3439

```
</btp:target-additional-information>
```

3440

```
<btp:decider-address>
```

3441

```
..address...
```

3442

```
</btp:decider-address>
```

3443

```

3444     <btp:transaction-identifier>...hexstring...</btp:transaction-
3445 identifier>
3446     <btp:status-list>
3447         <btp:status-item> +
3448             <btp:inferior-handle>...hexstring...</btp:inferior-
3449 handle>
3450             <btp:status>active|resigned|preparing|prepared|
3451                 autonomously-confirmed|autonomously-cancelled|
3452                 confirming|confirmed|cancelling|cancelled|
3453                 cancel-contradiction|confirm-contradiction|
3454                 hazard</btp:status>
3455             <btp:qualifiers> ?
3456                 ...qualifiers...
3457             </btp:qualifiers>
3458         </btp:status-item>
3459     </btp:status-list>
3460     <btp:qualifiers> ?
3461         ...qualifiers...
3462     </btp:qualifiers>
3463 </btp:inferior_statuses>
3464
3465

```

REQUEST_STATUS

```

3466
3467
3468     <btp:request_status id?>
3469         <btp:target-additional-information>
3470             ...additional address information...
3471         </btp:target-additional-information>
3472         <btp:reply-address>
3473             ...address...
3474         </btp:reply-address>
3475         <btp:inferior-identifier>...hexstring...</btp:inferior-
3476 identifier> ?
3477         <btp:transaction-identifier>...hexstring...</btp:transaction-
3478 identifier> ?
3479         <btp:qualifiers> ?
3480             ...qualifiers...
3481         </btp:qualifiers>
3482     </btp:request_status>
3483
3484

```

STATUS

```

3485
3486
3487     <btp:status id?>
3488         <btp:target-additional-information>
3489             ...additional address information...
3490         </btp:target-additional-information>
3491         <btp:inferior-address> ?
3492             ...address...
3493         </btp:inferior-address>
3494         <btp:inferior-identifier>...hexstring...</btp:inferior-
3495 identifier> ?
3496         <btp:decider-address> ?

```

```

3497     ...address...
3498     </btp:decider-address>
3499     <btp:transaction-identifier>...hexstring...</btp:transaction-
3500 identifier> ?
3501     <btp:status-value>created|enrolling|active|resigning|
3502         resigned|preparing|prepared|
3503         confirming|confirmed|cancelling|cancelled|
3504         cancel-contradiction|confirm-contradiction|
3505         hazard|contradicted|unknown|inaccessible</btp:status-
3506 value>
3507     <btp:qualifiers> ?
3508     ...qualifiers...
3509     </btp:qualifiers>
3510 </btp:status>
3511

```

REDIRECT

```

3513
3514
3515 <btp:redirect id?>
3516     <btp:target-additional-information>
3517         ...additional address information...
3518     </btp:target-additional-information>
3519     <btp:superior-identifier>...hexstring...</btp:superior-
3520 identifier> ?
3521     <btp:inferior-identifier>...hexstring...</btp:inferior-
3522 identifier>
3523     <btp:old-address> +
3524         ...address...
3525     </btp:old-address>
3526     <btp:new-address> +
3527         ...address...
3528     </btp:new-address>
3529     <btp:qualifiers> ?
3530     ...qualifiers...
3531     </btp:qualifiers>
3532 </btp:redirect>
3533
3534

```

FAULT

```

3535
3536
3537 <btp:fault id?>
3538     <btp:target-additional-information>
3539         ...additional address information...
3540     </btp:target-additional-information>
3541     <btp:superior-identifier>...hexstring...</btp:superior-
3542 identifier> ?
3543     <btp:inferior-identifier>...hexstring...</btp:inferior-
3544 identifier> ?
3545     <btp:fault-type>...fault type name...</btp:fault-type>
3546     <btp:fault-data>...fault data...</btp:fault-data> ?
3547     <btp:qualifiers> ?
3548     ...qualifiers...
3549     </btp:qualifiers>

```

3550 </btp:fault>

3551

3552

3553 The following fault type names are represented by simple strings, corresponding to the entries
3554 defined in the abstract message set:

3555

- 3556 o general
- 3557 o unknown-parameter
- 3558 o wrong-state
- 3559 o communication-failure
- 3560 o invalid-superior
- 3561 o duplicate-inferior
- 3562 o unknown-inferior

3563

3564 Revisions of this specification may add other fault type names, which shall be simple strings
3565 of letters, numbers and hyphens. If other specifications define fault type names to be used
3566 with BTP, the names shall be URIs.

3567

3568 Fault data can take on various forms:

3569

3570 Free text:

3571

3572 <btp:fault-data>...string data...</btp:fault-data>

3573

3574 Identifier:

3575

3576 <btp:fault-data>...hexstring...</btp:fault-data>

3577

3578

3579 Inferior Identity:

3580

```
3581 <btp:fault-data>
3582   <btp:inferior-address> +
3583     ...address...
3584   </btp:inferior-address>
3585   <btp:inferior-identifier>...hexstring...</btp:inferior-
3586   identifier>
3587 </btp:fault-data>
```

3588

3589

3590 **Standard qualifiers**

3591 The informal syntax for these messages assumes the namespace prefix “btpq” is associated
3592 with the URI “urn:oasis:names:tc:BTP:qualifiers”.

3593

3594 **Transaction timelimit**

3595

```
3596 <btpq:transaction-timelimit>
3597   <btpq:timelimit>
3598     ...time in seconds...
```

3598

```
3599     </btpq:timelimit>
3600 </btpq:transaction-timelimit>
3601
```

Inferior timeout

```
3602     <btpq:inferior-timeout>
3603     <btpq:timeout>
3604     ..time in seconds...
3605     </btpq:timeout>
3606     <btpq:intended-decision>confirm|cancel</btpq:intended-decision>
3607 </btpq:inferior-timeout>
3608
3609
```

Minimum inferior timeout

```
3610     <btpq:minimum-inferior-timeout>
3611     <btpq:minimum-timeout>
3612     ..time in seconds...
3613     </btpq:minimum-timeout>
3614 </btpq:minimum-inferior-timeout>
3615
3616
```

Compounding of Messages

3617
3618
3619 Relating BTP to one another, in a “group” is represented by containing them within the
3620 btp:relatedgroup element, with the related messages as child elements. The processing for the
3621 group is defined in the section “Groups – combinations of related messages”. For example

```
3622  
3623     <btp:relatedgroup>
3624     <btp:context-reply>
3625     ..<completion-status>related</completion-status> ...
3626     </btp:context-reply>
3627     <btp:enrol>...</btp:enrol>
3628     <btp:prepared>...</btp:prepared>
3629     </btp:relatedgroup>
3630
```

3631 The carrier protocol binding specifies how a relation between application and BTP messages
3632 is represented.

3633
3634 Bundling (semantically insignificant combination) of BTP messages and related groups is
3635 indicated with the "btp:messages" element, with the bundled messages and related groups as
3636 child elements. For example (confirming one and cancelling another inferiors of a cohesion):

```
3637     <btp:messages>
3638     <btp:enrolconfirm>...</btp:enrolconfirm>
3639     <btp:preparedcancel>...</btp:preparedcancel>
3640 </btp:messages>
3641
3642
```

3643
3644 Relating BTP messages to one another is achieved through containment. For example:

```
3645  
3646     <btp:context-reply>
3647     ..<completion-status>related</completion-status> ...
3648     <btp:enrol>...</btp:enrol>
```

3649
3650
3651
3652
3653

```
</btp:context-reply>
```

~~The carrier protocol binding specifies how a relation between application and BTP messages is represented.~~

3654

3655 **Carrier Protocol Bindings**

3656

3657 The notion of bindings is introduced to act as the glue between the BTP XML messages and
3658 an underlying transport. A binding specification must define various particulars of how the
3659 BTP messages are carried and some aspects of how the related application messages are
3660 carried. This document specifies two bindings: a SOAP binding and a SOAP + Attachments
3661 binding. However, other bindings could be specified by the Oasis BTP technical committee
3662 or by a third party. For example, in the future a binding might exist to put a BTP message
3663 directly on top of HTTP without the use of SOAP, or a closed community could define their
3664 own binding. To ensure that such specifications are complete, the Binding Proforma defines
3665 the information that must be included in a binding specification.
3666

3667 **Carrier Protocol Binding Proforma**

3668

3669 A BTP carrier binding specification should provide the following information:

3670

3671 **Binding name:** A name for the binding, as used in the “binding name” field of BTP
3672 addresses (and available for declaring the capabilities of an implementation). Binding
3673 specified in this document, and future revisions of this document have binding names that are
3674 simple strings of letters, numbers and hyphens (and, in particular, do not contain colons).
3675 Bindings specified elsewhere shall have binding names that are URIs. Bindings specified in
3676 this document use numbers to identify the version of the binding, not the version(s) of the
3677 carrier protocol.

3678

3679 **Binding address format:** This section states the format of the “binding address” field of a
3680 BTP address for this binding. For many bindings, this will be a URL of some kind; for other
3681 bindings it may be some other form

3682

3683 **BTP message representation:** This section will define how BTP messages are represented.
3684 For many bindings, this will be the normal string encoding of the XML, in accordance with
3685 the XML schema defined in this document.

3686

3687 **Mapping for BTP messages (unrelated) :** This section will define how BTP messages that
3688 are not related to application messages are sent in either direction between Superior and
3689 Inferior. (i.e. those messages sent directly between BTP actors). This mapping need not be
3690 symmetric (i.e. Superior to Inferior may differ to some degree to Inferior to Superior). The
3691 mapping may define particular rules for particular BTP messages, or messages with particular
3692 parameter values (e.g. the FAULT message with “fault-type” “CommunicationFailure” will
3693 typically not be sent as a BTP message). The mapping states any constraints or requirements
3694 on which BTP may or must be bundled together by compounding.

3695

3696 **Mapping for BTP messages related to application messages-** This section will define
3697 how BTP messages that are related to application messages are sent. A binding specification
3698 may defer details of this to a particular application (e.g. a mapping specification could just
3699 say “the CONTEXT may be carried as a parameter of an application invocation”).

3700 Alternatively, the binding may specify a general method that represents the relationship
3701 between application and BTP messages.

3702
3703 **Implicit messages:** This section specifies which BTP messages, if any, are not sent explicitly
3704 but are treated as implicit in application messages or other BTP messages. This may depend
3705 on particular parameter values of the BTP messages or the application messages.

3706
3707 **Faults:** The relationship between the fault and exception reporting mechanisms of the carrier
3708 protocol and of BTP shall be defined. This may include definition of which carrier protocol
3709 exceptions are equivalent to a FAULT/communication-failure message.

3710
3711 **Relationship to other bindings:** Any relationship to other bindings is defined in this section.
3712 If BTP addresses with different bindings are be considered to match (for purposes of
3713 identifying the peer Superior/Inferior and redirection), this should be specified here.

3714
3715 **Limitations on BTP use:** Any limitations on the full range of BTP functionality that are
3716 imposed by use of this binding should be listed. This would include limitations on which
3717 messages can be sent, which event sequences are supported and restrictions on parameter
3718 values. Such limitations may reduce the usefulness of an implementation, but may be
3719 appropriate in certain environments.

3720
3721 **Other:** Other features of the binding, especially any that will potentially affect interoperation
3722 should be specified here. This may include restrictions or requirements on the use or support
3723 of optional carrier parameters or mechanisms >.

3724

3725 Bindings for request/response carrier protocols

3726

3727 change justification – start from outcome and its spontaneous receipt; must handle on req;
3728 can avoid always using req if allow on rsp – so permissive

3729

3730 BTP does not generally follow request/response pattern. In particular, on the outcome
3731 relationship either side may initiate a message – this is an essential part of the presume-abort
3732 recovery paradigm although it is not limited to recovery cases. However, there are some BTP
3733 messages, especially in the control relationship, that do have a request/response pattern.
3734 Many (potential) carrier protocols (e.g. HTTP) do have a request/response pattern. A BTP
3735 binding specifications to a request/response carrier protocol needs to state what rules apply –
3736 which messages can be carried by requests, which by responses. The simplest rule is to send
3737 all BTP messages on requests, and let the carrier responses travel back empty. This would be
3738 inefficient in use of network resources, and possibly inconvenient when used for the BTP
3739 request/response pairs.

3740

3741 This section defines a set of rules that allow more efficient use of the carrier, while allowing
3742 the initiator of a BTP request/response pair to ensure the BTP response is sent back on the
3743 carrier response. These rules are specified in this section to enable binding specifications to
3744 reference them, without requiring each binding specification to repeat similar information.

3745

3746 A binding to a request/response carrier is not required to use these rules. It may define other
3747 rules.

3748

3749 Request/response exploitation rules

3750

3751 These rules allow implementations to use the request and response of the carrier protocol
3752 efficiently, and, when a BTP request/response exchange occurs, to either treat the
3753 request/response exchanges of the carrier protocol and of BTP independently, if both sides
3754 wish, or allow either side to map them closely.

3755

3756 Under these rules, an implementation sending a BTP request (i.e. a message, other than
3757 CONTEXT, which has “reply-address” as a parameter in the abstract message definition), can
3758 ensure that it and the reply map to a carrier request/response by supplying no value for the
3759 “reply-address”. An implementation receiving such a request is required to send the BTP
3760 response on the carrier response.

3761

3762 Conversely, if an implementation does supply a “reply-address” value on the request, the
3763 receiver has the option of sending the BTP response back on the carrier response, or sending
3764 it on a new carrier request.

3765

3766 Within the outcome relationship, apart from ENROL/ENROLLED, there is no “reply-
3767 address”, and the parties know each other’s “address-as-superior” and “address-as-inferior”.
3768 Both sides are permitted to treat the carrier request/response exchanges as just opportunities
3769 for sending messages to the appropriate destination.

3770

3771 The rules:

3772

3773 a) A BTP actor **may** bundle one or more BTP messages and related groups that
3774 have the same binding address for their target in a single btp:messages and
3775 transmit this btp:messages element on a carrier protocol request. There is no
3776 restriction on which combinations of messages and groups may be so bundled,
3777 other than that they have the same binding address, and that this binding address
3778 is usable as the destination of a carrier protocol request.

3779

3780 b) A BTP actor that has received a carrier protocol request to which it has not yet
3781 responded, and which has one or more BTP messages and groups whose binding
3782 address for the target matches the origin of the carrier request **may** bundle such
3783 BTP messages in a single btp:messages element and transmit that on the carrier
3784 protocol response.

3785

3786 c) A BTP actor that has received, on a carrier protocol request, one or more BTP
3787 messages or related groups that require a BTP response and for which no reply
3788 address was supplied, **must** bundle the responding BTP message and groups in a
3789 btp:messages element and transmit this element on the carrier protocol response
3790 to the request that carried the BTP request.

3791

- 3792 d) Where only one message or group is to be sent, it shall be contained within a
3793 btm:messages element, as a bundle of one element.
- 3794
- 3795 e) A BTP actor that receives a carrier protocol request carrying BTP messages that
3796 do have a reply address, or which initiate processing that produces BTP messages
3797 whose target binding address matches the origin of the request, may freely
3798 choose whether to use the carrier protocol response for the replies, or to send
3799 back an “empty carrier protocol response”, and send the BTP replies in a
3800 separately initiated carrier protocol request. The characteristics of an “empty
3801 carrier protocol response” shall be stated in the particular binding specification.
- 3802
- 3803 f) A BTP actor that sends BTP messages on a carrier protocol request must be able
3804 to accept returning BTP messages on the corresponding carrier protocol response
3805 and, if the actor has offered an address on which it will receive carrier requests,
3806 must be able to accept “replying” BTP messages on a separate carrier protocol
3807 request.
- 3808

3809 SOAP Binding

3810

3811 This binding describes how BTP messages will be carried using SOAP as in the [SOAP 1.1](#)
3812 specification. If no application message is sent at the same time, the BTP messages are
3813 contained within the SOAP-Body element. If application messages are sent, the BTP
3814 messages are contained in the SOAP-Header element. In the former case, an additional XML
3815 wrapper element is used to allow implementations to treat the SOAP messages as using the
3816 RPC convention, or to use the document convention

3817

3818 **Binding name:** soap-http-1

3819

3820 **Binding address format:** shall be a URL, of type HTTP.

3821

3822 **BTP message representation:** The string representation of the XML, as specified in the
3823 XML schema defined in this document shall be used. BTP messages conform to the rules of
3824 the Section 5 (of the SOAP 1.1 specification) SOAP Encoding as specified by the URI:
3825 "http://schemas.xmlsoap.org/soap/encoding/".

3826

3827 **Mapping for BTP messages (unrelated):** The “request/response exploitation” rules shall be used.

3828

3829 When BTP messages are sent on an HTTP request which is not carrying an application message,
3830 the messages are contained in a single btm:messages element which is the immediate child
3831 element of a btm:transmission element which is the immediate child element of the SOAP-
3832 Body element. When BTP messages are sent on an HTTP response that is the reply to an
3833 HTTP request containing (in the SOAP-Body) a btm:transmission element, the BTP messages
3834 are contained in a single btm:messages element which is the immediate child element of a
3835 btm:transmissionResponse element which is the immediate child element of the SOAP-Body
3836 element.

3837

3838 An “empty carrier protocol response” to be sent when an HTTP request containing a
3839 btpt:transmission element has been received and the request/response exploitation rules allow
3840 an empty carrier protocol response and the implementation BTP actor chooses just reply at
3841 the lower level, shall be any of:

- 3842 a) an empty HTTP response
- 3843 b) an HTTP response containing an empty SOAP-Envelope
- 3844 c) an HTTP response containing a SOAP-Envelope containing a
3845 btpt:transmissionResponse element containing a single, empty btpt:messages
3846 element.

3847
3848 The receiver (the initial sender of the HTTP request) shall treat these in the same way – they
3849 have no effect on the BTP sequence (other than indicating that the earlier sending did not
3850 cause a communication failure.)

3851 Note – The transmission and transmissionResponse elements allows
3852 implementations to treat the transmission of BTP messages (when no
3853 application message is present) as either a SOAP “rpc” invocation, or as a
3854 SOAP “document” (literal) transfer. Implementations that take opposite
3855 approaches can interwork directly and freely. As an RPC invocation, a
3856 method called “transmission” is invoked, with one argument (the
3857 btpt:messages), and an optional return value (btpt:messages). As a document
3858 transfer, the btpt:transmission and btpt:transmission-response wrappers do not
3859 imply any semantics.

3860 ~~If no application message is being sent at the same time, BTP messages shall be contained in~~
3861 ~~a btpt:messages element which shall be an immediate child element of the SOAP-Body. There~~
3862 ~~shall be precisely one btpt:messages element. Any number of BTP messages with the same~~
3863 ~~binding address in their target address may be carried in the same btpt:messages element.~~
3864

3865 If an application message is being sent at the same time, the mapping for related messages
3866 shall be used, as if the BTP messages were related to the application message. (There is no
3867 ambiguity in whether the BTP messages are related, because only CONTEXT and ENROL
3868 can be related to an application message.)
3869

3870 **Mapping for BTP messages related to application messages:** All BTP messages sent with
3871 an application message, whether related to the application message or not, shall be sent in a
3872 single btpt:messages element in the SOAP:Header. There shall be precisely one btpt:messages
3873 element in the SOAP:Header.

3874 The “request/response exploitation” rules shall apply to the BTP messages carried in the
3875 SOAP:Header, as if they had been carried in a SOAP:Body, unrelated to an application
3876 message, sent to the same binding address.

3877 Note – The specification of the application protocol will determine whether
3878 an RPC or document approach is used.

3879 Only CONTEXT and ENROL messages are related (&) to application messages. If there is
3880 only one CONTEXT or one ENROL message present in the SOAP-Body, and it does not

3881 have an XML ID attribute, it is related to the whole of the application message in the SOAP-
3882 Body. If a CONTEXT or ENROL message has an XML ID attribute, it is related only to
3883 those parts of the application message that reference it (using an XML REF attribute). There
3884 can be multiple such CONTEXT or ENROL messages, each with an XML ID attribute. If
3885 there are multiple CONTEXT or ENROL messages and any do not have an XML ID
3886 attribute, such message are not related any of the application message in the SOAP-Body.
3887

3888 Note -- Whether the relatedness has any significance for the application
3889 (particularly in the case of ENROL, without an ID parameter, carried with a
3890 response), is a matter for the application.

3891
3892 **Implicit messages:** A SOAP fault, or other communication failure received in response to a
3893 SOAP request that had a CONTEXT in the SOAP:Header shall be treated as if a
3894 CONTEXT_REPLY/repudiated had been received. See also the discussion under “other”
3895 about the SOAP mustUnderstand attribute.

3896
3897 **Faults:** A SOAP fault or other communication failure shall be treated as
3898 FAULT/communication-failure.
3899

3900 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
3901 string “soap-http-1” is considered to match one that has the binding string “soap-attachments-
3902 http-1” if the binding address and additional information fields match.
3903

3904 **Limitations on BTP use:** None
3905

3906 **Other:** The SOAP BTP binding does not make use of SOAPAction HTTP header or actor
3907 attribute. The SOAPAction HTTP header is left to be application specific when there are
3908 application messages in the SOAP:Body, as an already existing web service that is being
3909 upgraded to use BTP might have already made use of SOAPAction. The SOAPAction HTTP
3910 header shall be omitted when the SOAP message carries only BTP messages in the
3911 SOAP:Body.
3912

3913 The SOAP mustUnderstand attribute, when used on the btp:messages containing a the BTP
3914 CONTEXT, ensures that the server (as a whole) determines whether any enrolments are
3915 necessary and reply with CONTEXT_REPLY as appropriate. If mustUnderstand if false, a
3916 server can ignore the CONTEXT (if BTP is not supported there). It is an implementation or
3917 configuration option whether a CONTEXT_REPLY/ok is assumed to be implicit in the HTTP
3918 response in such a case. (If no CONTEXT_REPLY/ok is assumed, it will be impossible for
3919 the business transaction to confirm) .
3920

3921 Note – some SOAP implementations may not support the mustUnderstand
3922 attribute sufficiently to enforce these requirements. If using such an
3923 implementation on the service side, it may be necessary to assume an
3924 CONTEXT_REPLY/ok.

3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975

Example scenario using SOAP binding

The example below shows an application request with CONTEXT message sent from client.example.com (which includes the Superior) to services.example.com (Service).

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  soap-
env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

  <soap:Header>

    <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
      <btp:context superior-type="atom">
        <btp:superior-address>
          <btp:binding>soap-http-1</btp:binding>
          <btp:binding-
address>http://client.example.com/soaphandler</btp:binding-
address>
          <btp:additional-information>btpengine</btp:additional-
information>
          </btp:superior-address>
          <btp:superior-identifier>1001</btp:superior-identifier>
          <btp:qualifiers>
            <btpq:transaction-timelimit
xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers">1800</btpq:transact
ion-timelimit>
            </btp:qualifiers>
          </btp:context>
        </btp:messages>

      </soap:Header>

      <soap:Body>

        <ns1:orderGoods
xmlns:ns1="http://example.com/2001/Services/xyzgoods">
          <custID>ABC8329045</custID>
          <itemID>224352</itemID>
          <quantity>5</quantity>
        </ns1:orderGoods>

      </soap:Body>

    </soap:Envelope>
```

The example below shows CONTEXT_REPLY and a related (and therefore contained) ENROL message sent from services.example.com to client.example.com, in reply to the

3976 previous message. There is no application response, so the BTP messages are in the
3977 SOAP:Body.

```
3978 <soap:Envelope
3979     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3980     soap-
3981     env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
3982
3983     <soap:Header>
3984     </soap:Header>
3985
3986     <soap:Body>
3987
3988         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
3989             <btp:context-reply>
3990                 <btp:superior-address>
3991                     <btp:binding>soap-http-1</btp:binding>
3992                     <btp:binding-address>
3993                         http://client.example.com/soaphandler
3994                     </btp:binding-address>
3995                     <btp:additional-information>
3996                         btpengine
3997                     </btp:additional-information>
3998                 </btp:superior-address>
3999                 <btp:superior-identifier>1001</btp:superior-identifier>
4000                 <completion-status>related</completion-status>
4001
4002                 <btp:enrol reply-requested="false">
4003                     <btp:target-additional-information>
4004                         btpengine
4005                     </btp:target-additional-information>
4006                     <btp:superior-identifier>
4007                         1001
4008                     </btp:superior-identifier>
4009                     <btp:inferior-address>
4010                         <btp:binding>soap-http-1</btp:binding>
4011                         <btp:binding-address>
4012                             http://services.example.com/soaphandler
4013                         </btp:binding-address>
4014                     </btp:inferior-address>
4015                     <btp:inferior-identifier>
4016                         AAAB
4017                     </btp:inferior-identifier>
4018                 </btp:enrol>
4019             </btp:context-reply>
4020         </btp:messages>
4021     </soap:Body>
4022 </soap:Envelope>
```

4029
4030

4031 **SOAP + Attachments Binding**
4032

4033 This binding describes how BTP messages will be carried using SOAP as in the [SOAP](#)
4034 [Messages with Attachments](#) specification. It is a superset of the Basic SOAP binding, soap-
4035 http-1. The two bindings only differ when application messages are sent
4036

4037 **Binding name:** soap-attachments-http-1

4038 **Binding address format:** as for soap-http-1
4039

4040 **BTP message representation:** As for soap-http-1
4041

4042 **Mapping for BTP messages (unrelated):** As for “soap-http-1”, except the SOAP:Envelope
4043 containing the SOAP-Body containing the BTP messages shall be in a MIME body part, as
4044 specified in [SOAP Messages with Attachments](#) specification. If an application message is
4045 being sent at the same time, the mapping for related messages for this binding shall be used,
4046 as if the BTP messages were related to the application message(s).
4047

4048 **Mapping for BTP messages related to application messages:** MIME packaging shall be
4049 used. One of the MIME multipart/related parts shall contain a SOAP:Envelope, whose
4050 SOAP:Headers element shall contain precisely one btp:messages element, containing any
4051 BTP messages. Any BTP CONTEXT in the btp:messages is considered to be related to the
4052 application message(s) in the SOAP:Body, and to also any of the MIME parts referenced
4053 from the SOAP:Body (using the “href” attribute).
4054

4055 **Implicit messages:** As for soap-http-1.
4056

4057 **Faults:** As for soap-http-1.
4058

4059 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
4060 string “soap-http-1” is considered to match one that has the binding string “soap-
4061 attachments-http-1” if the binding address and additional information fields match.
4062

4063 **Limitations on BTP use:** None
4064

4065 **Other:** As for soap-http-1
4066

4067 *Example using SOAP + Attachments binding*
4068

```
4069 MIME-Version: 1.0  
4070 Content-Type: Multipart/Related; boundary=MIME_boundary;  
4071 type=text/xml;  
4072     start="someID"  
4073  
4074 --MIME_boundary  
4075
```

```

4076 Content-Type: text/xml; charset=UTF-8
4077 Content-ID: someID
4078
4079 <?xml version='1.0' ?>
4080 <soap:Envelope
4081     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
4082     soap-
4083 env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
4084
4085     <soap:Header>
4086
4087         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
4088             <btp:context superior-type="atom">
4089                 <btp:superior-address>
4090                     <btp:binding>soap-http-1</btp:binding>
4091                     <btp:binding-address>
4092                         http://client.example.com/soaphandler
4093                     </btp:binding-address>
4094                 </btp:superior-address>
4095                 <btp:superior-identifier>1001</btp:superior-identifier>
4096             </btp:context>
4097         </btp:messages>
4098
4099     </soap:Header>
4100
4101     <soap:Body>
4102         <orderGoods href="cid:anotherID"/>
4103     </soap:Body>
4104
4105 </soap:Envelope>
4106
4107 --MIME_boundary
4108 Content-Type: text/xml
4109 Content-ID: anotherID
4110
4111     <ns1:orderGoods
4112 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
4113         <custID>ABC8329045</custID>
4114         <itemID>224352</itemID>
4115         <quantity>5</quantity>
4116     </ns1:orderGoods>
4117
4118
4119 --MIME_boundary--
4120
4121

```

XML Schema for SOAP Bindings

```

4122
4123
4124 <?xml version="1.0"?>
4125 <schema targetNamespace="urn:oasis:names:tc:BTP:xml"
4126     xmlns="http://www.w3.org/2001/XMLSchema"
4127     xmlns:tns="urn:oasis:names:tc:BTP:xml">

```

```

4128
4129     <complexType name="qualifier_type">
4130         <simpleContent>
4131             <extension base="string">
4132                 <attribute name="must-be-understood" type="boolean"/>
4133                 <attribute name="to-be-propagated" type="boolean"/>
4134             </extension>
4135         </simpleContent>
4136     </complexType>
4137     <element name="qualifier" type="tns:qualifier_type"/>
4138     <element name="qualifiers">
4139         <complexType>
4140             <sequence>
4141                 <element ref="tns:qualifier" maxOccurs="unbounded"/>
4142             </sequence>
4143         </complexType>
4144     </element>
4145
4146     <complexType name="address">
4147         <sequence>
4148             <element name="binding-name" type="string"/>
4149             <element name="binding-address" type="string"/>
4150             <element name="additional-information" type="string"
4151 minOccurs="0"/>
4152         </sequence>
4153     </complexType>
4154
4155     <simpleType name="identifier">
4156         <restriction base="string">
4157             <pattern value="([0-9,A-Z])*"/>
4158         </restriction>
4159     </simpleType>
4160
4161     <simpleType name="superior-type">
4162         <restriction base="string">
4163             <enumeration value="cohesion"/>
4164             <enumeration value="atom"/>
4165         </restriction>
4166     </simpleType>
4167
4168     <simpleType name="transaction-type">
4169         <restriction base="string">
4170             <enumeration value="cohesion"/>
4171             <enumeration value="atom"/>
4172         </restriction>
4173     </simpleType>
4174
4175
4176     <element name="context">
4177         <complexType>
4178             <sequence>
4179                 <element name="superior-address" type="tns:address"
4180 maxOccurs="unbounded"/>

```

```

4181         <element name="superior-identifier"
4182 type="tns:identifier"/>
4183         <element ref="tns:qualifiers" minOccurs="0"/>
4184     </sequence>
4185     <attribute name="id" type="ID" use="optional"/>
4186     <attribute name="superior-type" type="tns:superior-type"
4187 use="required"/>
4188 </complexType>
4189 </element>
4190
4191     <element name="context-reply">
4192     <complexType>
4193     <sequence>
4194         <element name="superior-address" type="tns:address"
4195 maxOccurs="unbounded"/>
4196         <element name="superior-identifier"
4197 type="tns:identifier"/>
4198         <element name="completion-status">
4199         <simpleType>
4200             <restriction base="string">
4201                 <enumeration value="completed"/>
4202                 <enumeration value="related"/>
4203                 <enumeration value="repudiated"/>
4204             </restriction>
4205         </simpleType>
4206         </element>
4207         <element ref="tns:qualifiers" minOccurs="0"/>
4208     </sequence>
4209     <attribute name="id" type="ID"/>
4210     <attribute name="superior-type" type="tns:superior-type"
4211 use="required"/>
4212 </complexType>
4213 </element>
4214
4215     <element name="begin">
4216     <complexType>
4217     <sequence>
4218         <element name="target-additional-information"
4219 type="string"/>
4220         <element name="reply-address" type="tns:address"/>
4221         <element ref="tns:qualifiers" minOccurs="0"/>
4222     </sequence>
4223     <attribute name="id" type="ID"/>
4224     <attribute name="transaction-type" type="tns:superior-type"
4225 use="required"/>
4226 </complexType>
4227 </element>
4228
4229     <element name="begun">
4230     <complexType>
4231     <sequence>
4232         <element name="target-additional-information"
4233 type="string"/>

```

```

4234         <element name="decider-address" type="tns:address"
4235 minOccurs="0" />
4236         <element name="transaction-identifier"
4237 type="tns:identifier" minOccurs="0" />
4238         <element name="inferior-handle" type="tns:identifier"
4239 minOccurs="0" />
4240         <element name="inferior-address" type="tns:address"
4241 minOccurs="0" />
4242         <element ref="tns:qualifiers" minOccurs="0" />
4243     </sequence>
4244     <attribute name="id" type="ID" />
4245     <attribute name="transaction-type" type="tns:superior-type"
4246 use="required" />
4247 </complexType>
4248 </element>
4249
4250     <element name="enrol">
4251         <complexType>
4252             <sequence>
4253                 <element name="target-additional-information"
4254 type="string" />
4255                 <element name="superior-identifier"
4256 type="tns:identifier" />
4257                 <element name="reply-address" type="tns:address"
4258 minOccurs="0" />
4259                 <element name="inferior-address" type="tns:address"
4260 minOccurs="1" maxOccurs="unbounded" />
4261                 <element name="inferior-identifier"
4262 type="tns:identifier" />
4263                 <element ref="tns:qualifiers" minOccurs="0" />
4264             </sequence>
4265             <attribute name="id" type="ID" />
4266             <attribute name="reply-requested" type="boolean" />
4267         </complexType>
4268     </element>
4269
4270
4271     <element name="enrolled">
4272         <complexType>
4273             <sequence>
4274                 <element name="target-additional-information"
4275 type="string" />
4276                 <element name="inferior-identifier"
4277 type="tns:identifier" />
4278                 <element name="inferior-handle" type="tns:identifier"
4279 minOccurs="0" />
4280                 <element ref="tns:qualifiers" minOccurs="0" />
4281             </sequence>
4282             <attribute name="id" type="ID" />
4283         </complexType>
4284     </element>
4285
4286     <element name="resign">

```

```

4287     <complexType>
4288         <sequence>
4289             <element name="target-additional-information"
4290 type="string"/>
4291             <element name="superior-identifier"
4292 type="tns:identifier"/>
4293             <element name="inferior-address" type="tns:address"
4294 minOccurs="1" maxOccurs="unbounded"/>
4295             <element name="inferior-identifier"
4296 type="tns:identifier"/>
4297             <element ref="tns:qualifiers" minOccurs="0"/>
4298         </sequence>
4299         <attribute name="id" type="ID"/>
4300         <attribute name="response-requested" type="boolean"/>
4301     </complexType>
4302 </element>
4303
4304     <element name="resigned">
4305         <complexType>
4306             <sequence>
4307                 <element name="target-additional-information"
4308 type="string"/>
4309                 <element name="inferior-identifier"
4310 type="tns:identifier"/>
4311                 <element ref="tns:qualifiers" minOccurs="0"/>
4312             </sequence>
4313             <attribute name="id" type="ID"/>
4314         </complexType>
4315     </element>
4316
4317     <element name="prepare">
4318         <complexType>
4319             <sequence>
4320                 <element name="target-additional-information"
4321 type="string"/>
4322                 <element name="inferior-identifier"
4323 type="tns:identifier" minOccurs="0"/>
4324                 <element name="reply-address" type="tns:address"
4325 minOccurs="0"/>
4326                 <element name="transaction-identifier"
4327 type="tns:identifier" minOccurs="0"/>
4328                 <element name="inferiors-list" minOccurs="0">
4329                     <complexType>
4330                         <sequence>
4331                             <element name="inferior-handle"
4332 type="tns:identifier" maxOccurs="unbounded"/>
4333                         </sequence>
4334                     </complexType>
4335                 </element>
4336                 <element ref="tns:qualifiers" minOccurs="0"/>
4337             </sequence>
4338             <attribute name="id" type="ID"/>
4339         </complexType>

```

```

4340     </element>
4341
4342     <element name="prepared">
4343         <complexType>
4344             <sequence>
4345                 <element name="target-additional-information"
4346 type="string" />
4347                 <element name="superior-identifier"
4348 type="tns:identifier" />
4349                 <element name="inferior-address" type="tns:address"
4350 maxOccurs="unbounded" />
4351                 <element name="inferior-identifier"
4352 type="tns:identifier" />
4353                 <element ref="tns:qualifiers" minOccurs="0" />
4354             </sequence>
4355             <attribute name="id" type="ID" />
4356             <attribute name="default-is-cancel" type="boolean" />
4357         </complexType>
4358     </element>
4359
4360     <element name="confirm">
4361         <complexType>
4362             <sequence>
4363                 <element name="target-additional-information"
4364 type="string" />
4365                 <element name="inferior-identifier"
4366 type="tns:identifier" />
4367                 <element ref="tns:qualifiers" minOccurs="0" />
4368             </sequence>
4369             <attribute name="id" type="ID" />
4370         </complexType>
4371     </element>
4372
4373     <element name="confirmed">
4374         <complexType>
4375             <sequence>
4376                 <element name="target-additional-information"
4377 type="string" />
4378                 <element name="superior-identifier"
4379 type="tns:identifier" />
4380                 <element name="inferior-address" type="tns:address"
4381 minOccurs="0" />
4382                 <element name="inferior-identifier"
4383 type="tns:identifier" minOccurs="0" />
4384                 <element name="decider-address" type="tns:address"
4385 minOccurs="0" />
4386                 <element name="transaction-identifier"
4387 type="tns:identifier" minOccurs="0" />
4388                 <element ref="tns:qualifiers" minOccurs="0" />
4389             </sequence>
4390             <attribute name="id" type="ID" />
4391             <attribute name="confirmed-received" type="boolean" />
4392         </complexType>

```

```

4393     </element>
4394
4395     <element name="cancel">
4396         <complexType>
4397             <sequence>
4398                 <element name="target-additional-information"
4399 type="string"/>
4400                 <element name="inferior-identifier"
4401 type="tns:identifier" minOccurs="0"/>
4402                 <element name="reply-address" type="tns:address"
4403 minOccurs="0"/>
4404                 <element name="transaction-identifier"
4405 type="tns:identifier" minOccurs="0"/>
4406                 <element name="decider-address" type="tns:address"
4407 minOccurs="0"/>
4408                 <element name="transaction-identifier"
4409 type="tns:identifier" minOccurs="0"/>
4410                 <element name="inferiors-list" minOccurs="0">
4411                     <complexType>
4412                         <sequence>
4413                             <element name="inferior-handle"
4414 type="tns:identifier" maxOccurs="unbounded"/>
4415                         </sequence>
4416                     </complexType>
4417                 </element>
4418                 <element ref="tns:qualifiers" minOccurs="0"/>
4419             </sequence>
4420             <attribute name="id" type="ID"/>
4421         </complexType>
4422     </element>
4423
4424     <element name="cancelled">
4425         <complexType>
4426             <sequence>
4427                 <element name="target-additional-information"
4428 type="string"/>
4429                 <element name="superior-identifier"
4430 type="tns:identifier"/>
4431                 <element name="inferior-address" type="tns:address"
4432 maxOccurs="unbounded"/>
4433                 <element name="inferior-identifier"
4434 type="tns:identifier" minOccurs="0"/>
4435                 <element name="decider-address" type="tns:address"
4436 minOccurs="0"/>
4437                 <element name="transaction-identifier"
4438 type="tns:identifier" minOccurs="0"/>
4439                 <element ref="tns:qualifiers" minOccurs="0"/>
4440             </sequence>
4441             <attribute name="id" type="ID"/>
4442         </complexType>
4443     </element>
4444
4445     <element name="hazard">

```

```

4446     <complexType>
4447         <sequence>
4448             <element name="target-additional-information"
4449 type="string"/>
4450             <element name="superior-identifier"
4451 type="tns:identifier"/>
4452             <element name="inferior-address" type="tns:address"
4453 maxOccurs="unbounded"/>
4454             <element name="inferior-identifier"
4455 type="tns:identifier"/>
4456             <element ref="tns:qualifiers" minOccurs="0"/>
4457         </sequence>
4458         <attribute name="id" type="ID"/>
4459     </complexType>
4460 </element>
4461
4462     <element name="contradiction">
4463         <complexType>
4464             <sequence>
4465                 <element name="target-additional-information"
4466 type="string"/>
4467                 <element name="inferior-identifier"
4468 type="tns:identifier"/>
4469                 <element ref="tns:qualifiers" minOccurs="0"/>
4470             </sequence>
4471             <attribute name="id" type="ID"/>
4472         </complexType>
4473     </element>
4474
4475     <element name="superior-state">
4476         <complexType>
4477             <sequence>
4478                 <element name="target-additional-information"
4479 type="string"/>
4480                 <element name="inferior-identifier"
4481 type="tns:identifier"/>
4482                 <element name="status">
4483                     <simpleType>
4484                         <restriction base="string">
4485                             <enumeration value="active"/>
4486                             <enumeration value="prepared-received"/>
4487                             <enumeration value="inaccessible"/>
4488                             <enumeration value="unknown"/>
4489                         </restriction>
4490                     </simpleType>
4491                 </element>
4492                 <element ref="tns:qualifiers" minOccurs="0"/>
4493             </sequence>
4494             <attribute name="id" type="ID"/>
4495             <attribute name="reply-requested" type="boolean"/>
4496         </complexType>
4497     </element>
4498

```

```

4499     <element name="inferior-state">
4500         <complexType>
4501             <sequence>
4502                 <element name="target-additional-information"
4503 type="string"/>
4504                 <element name="superior-identifier"
4505 type="tns:identifier"/>
4506                 <element name="inferior-address" type="tns:address"
4507 maxOccurs="unbounded"/>
4508                 <element name="inferior-identifier"
4509 type="tns:identifier"/>
4510                 <element name="status">
4511                     <simpleType>
4512                         <restriction base="string">
4513                             <enumeration value="active"/>
4514                             <enumeration value="prepared-received"/>
4515                             <enumeration value="inaccessible"/>
4516                             <enumeration value="unknown"/>
4517                         </restriction>
4518                     </simpleType>
4519                 </element>
4520                 <element ref="tns:qualifiers" minOccurs="0"/>
4521             </sequence>
4522             <attribute name="id" type="ID"/>
4523             <attribute name="reply-requested" type="boolean"/>
4524         </complexType>
4525     </element>
4526
4527     <element name="confirm-one-phase">
4528         <complexType>
4529             <sequence>
4530                 <element name="target-additional-information"
4531 type="string"/>
4532                 <element name="inferior-identifier"
4533 type="tns:identifier"/>
4534                 <element ref="tns:qualifiers" minOccurs="0"/>
4535             </sequence>
4536             <attribute name="id" type="ID"/>
4537             <attribute name="report-hazard" type="boolean"/>
4538         </complexType>
4539     </element>
4540
4541     <element name="request-confirm">
4542         <complexType>
4543             <sequence>
4544                 <element name="target-additional-information"
4545 type="string"/>
4546                 <element name="reply-address" type="tns:address"/>
4547                 <element name="transaction-identifier"
4548 type="tns:identifier"/>
4549                 <element name="inferiors-list" minOccurs="0">
4550                     <complexType>
4551                         <sequence>

```

```

4552         <element name="inferior-handle"
4553 type="tns:identifier" maxOccurs="unbounded" />
4554         </sequence>
4555     </complexType>
4556 </element>
4557     <element ref="tns:qualifiers" minOccurs="0" />
4558 </sequence>
4559     <attribute name="id" type="ID" />
4560     <attribute name="report-hazard" type="boolean" />
4561 </complexType>
4562 </element>
4563
4564     <element name="request-statuses">
4565         <complexType>
4566             <sequence>
4567                 <element name="target-additional-information"
4568 type="string" />
4569                 <element name="reply-address" type="tns:address" />
4570                 <element name="transaction-identifier"
4571 type="tns:identifier" />
4572                 <element name="inferiors-list" minOccurs="0">
4573                     <complexType>
4574                         <sequence>
4575                             <element name="inferior-handle"
4576 type="tns:identifier" maxOccurs="unbounded" />
4577                             </sequence>
4578                         </complexType>
4579                     </element>
4580                     <element ref="tns:qualifiers" minOccurs="0" />
4581                 </sequence>
4582                 <attribute name="id" type="ID" />
4583             </complexType>
4584         </element>
4585
4586     <element name="inferior-statuses">
4587         <complexType>
4588             <sequence>
4589                 <element name="target-additional-information"
4590 type="string" />
4591                 <element name="decider-address" type="tns:address" />
4592                 <element name="transaction-identifier"
4593 type="tns:identifier" />
4594                 <element name="status-list">
4595                     <complexType>
4596                         <sequence>
4597                             <element name="status-item" maxOccurs="unbounded">
4598                                 <complexType>
4599                                     <sequence>
4600                                         <element name="inferior-handle"
4601 type="tns:identifier" />
4602                                         <element name="status">
4603                                             <simpleType>
4604                                                 <restriction base="string">

```

```

4605         <enumeration value="active" />
4606         <enumeration value="resigned" />
4607         <enumeration value="preparing" />
4608         <enumeration value="prepared" />
4609         <enumeration value="autonomously-
4610 confirmed" />
4611         <enumeration value="autonomously-
4612 cancelled" />
4613         <enumeration value="confirming" />
4614         <enumeration value="confirmed" />
4615         <enumeration value="cancelling" />
4616         <enumeration value="cancelled" />
4617         <enumeration value="cancel-contradiction" />
4618         <enumeration value="confirm-contradiction" />
4619         <enumeration value="hazard" />
4620     </restriction>
4621 </simpleType>
4622 </element>
4623 <element ref="tns:qualifiers" minOccurs="0" />
4624 </sequence>
4625 </complexType>
4626 </element>
4627 </sequence>
4628 </complexType>
4629 </element>
4630 <element ref="tns:qualifiers" minOccurs="0" />
4631 </sequence>
4632 <attribute name="id" type="ID" />
4633 </complexType>
4634 </element>
4635
4636 <element name="request-status">
4637 <complexType>
4638 <sequence>
4639 <element name="target-additional-information"
4640 type="string" />
4641 <element name="reply-address" type="tns:address" />
4642 <element name="inferior-identifier"
4643 type="tns:identifier" minOccurs="0" />
4644 <element name="transaction-identifier"
4645 type="tns:identifier" minOccurs="0" />
4646 <element ref="tns:qualifiers" minOccurs="0" />
4647 </sequence>
4648 <attribute name="id" type="ID" />
4649 </complexType>
4650 </element>
4651
4652 <element name="status">
4653 <complexType>
4654 <sequence>
4655 <element name="target-additional-information"
4656 type="string" />

```

```

4657         <element name="inferior-address" type="tns:address"
4658 minOccurs="0" />
4659         <element name="inferior-identifier"
4660 type="tns:identifier" minOccurs="0" />
4661         <element name="decider-address" type="tns:address"
4662 minOccurs="0" />
4663         <element name="transaction-identifier"
4664 type="tns:identifier" minOccurs="0" />
4665         <element name="status-value">
4666             <simpleType>
4667                 <restriction base="string">
4668                     <enumeration value="created" />
4669                     <enumeration value="enrolling" />
4670                     <enumeration value="active" />
4671                     <enumeration value="resigning" />
4672                     <enumeration value="resigned" />
4673                     <enumeration value="preparing" />
4674                     <enumeration value="prepared" />
4675                     <enumeration value="confirming" />
4676                     <enumeration value="confirmed" />
4677                     <enumeration value="cancelling" />
4678                     <enumeration value="cancelled" />
4679                     <enumeration value="cancel-contradiction" />
4680                     <enumeration value="confirm-contradiction" />
4681                     <enumeration value="hazard" />
4682                     <enumeration value="contradicted" />
4683                     <enumeration value="unknown" />
4684                     <enumeration value="inaccessible" />
4685                 </restriction>
4686             </simpleType>
4687         </element>
4688         <element ref="tns:qualifiers" minOccurs="0" />
4689     </sequence>
4690     <attribute name="id" type="ID" />
4691 </complexType>
4692 </element>
4693
4694     <element name="redirect">
4695         <complexType>
4696             <sequence>
4697                 <element name="target-additional-information"
4698 type="string" />
4699                 <element name="superior-identifier"
4700 type="tns:identifier" minOccurs="0" />
4701                 <element name="inferior-identifier"
4702 type="tns:identifier" />
4703                 <element name="old-address" type="tns:address"
4704 maxOccurs="unbounded" />
4705                 <element name="new-address" type="tns:address"
4706 maxOccurs="unbounded" />
4707                 <element ref="tns:qualifiers" minOccurs="0" />
4708             </sequence>
4709     <attribute name="id" type="ID" />

```

```

4710     </complexType>
4711 </element>
4712
4713     <element name="fault">
4714         <complexType>
4715             <sequence>
4716                 <element name="target-additional-information"
4717 type="string"/>
4718                 <element name="superior-identifier"
4719 type="tns:identifier" minOccurs="0"/>
4720                 <element name="inferior-identifier"
4721 type="tns:identifier" minOccurs="0"/>
4722                 <element name="fault-type" type="string"/>
4723                 <element name="fault-data" type="anyType"
4724 minOccurs="0"/>
4725                 <element ref="tns:qualifiers" minOccurs="0"/>
4726             </sequence>
4727             <attribute name="id" type="ID"/>
4728         </complexType>
4729     </element>
4730
4731 </schema>
4732

```

4732
4733

4734 **Conformance**

4735
4736
4737
4738
4739
4740

A BTP implementation need not implement all aspects of the protocol to be useful. The level of conformance of an implementation is defined by which roles it can support using the specified messages and carrier protocol bindings for interoperation with other implementations.

4741
4742
4743
4744
4745

A partially conformant implementation may implement some roles in a non-interoperable way, giving that implementation's users comparable proprietary functionality.

The following Roles and Role Groups are used to define conformance:

Role Group	Role
Initiator/Terminator	Initiator Terminator
Cohesive Hub	Factory Composer (as Decider and Superior) Coordinator (as Decider and Superior) Sub-composer Sub-coordinator
Atomic Hub	Factory Coordinator Sub-coordinator
Cohesive Superior	Composer (as Superior only) Sub-Composer Coordinator (as Superior only) Sub-coordinator
Atomic Superior	Coordinator (as Superior only)) Sub-coordinator
Participant	Inferior

Enroller

4746
4747
4748
4749
4750

An implementation may support one or more Role Groups. The following combinations are defined as commonly expected conformance profiles, although other combinations or selections are equally possible.

Conformance Profile	Role Groups
Participant Only	Participant
Atomic	Atomic Superior Participant
Cohesive	Full Superior Participant
Atomic Coordination Hub	Initiator/Terminator Atomic Coordination Hub Participant
Cohesive Coordination Hub	Initiator/Terminator Cohesive Coordination Hub Participant

4751
4752
4753
4754
4755
4756
4757
4758
4759
4760

BTP has several features, such as optional parameters, that allow alternative implementation architectures. Implementations should pay particular attention to avoid assuming their peers have made the same implementation options as they have (e.g. an implementation that always sends ENROL with the same inferior address and with the reply address absent (because the Inferior in all transactions are dealt with by the same addressable entity), must not assume that the same is true of received ENROLs)

4760 Part 3. Appendices

4761

4762

4763

These terms seem to be all either not used, or effectively defined elsewhere

4764 **A. Glossary**

4765

Message	A datum which is produced and then consumed.
Sender	The producer of a message.
Receiver	The consumer of a message.
Transmission	The passage of a message from a sender to a receiver.
Endpoint	A sender or receiver.
Address	An identifier for an endpoint.
Carrier Protocol	A protocol which defines how transmissions occur.
Carrier Protocol Address (CPA)	The address of an endpoint for a particular carrier protocol.
Business Transaction Protocol Address (BTPA)	A compound address consisting of a mandatory <i>carrier protocol address</i> and an optional opaque suffix.
	<i>PRF - suffix ? I've used "additional information"</i>
Actor	An entity which executes procedures, a software agent.
Application	An actor which uses the Business Transaction Protocol.
Application Message	A message produced by an application and consumed by an application.
Application Endpoint	An endpoint of an application message.
Operation	A procedure which is started by a receiver when a message arrives at it

	message arrives at it.
Application Operation	An operation which is started when an application message arrives.
Contract	Any rule, agreement or promise which constrains an actor's behaviour and is known to any other actor, and upon which any other knowing actor may rely.
Appropriate	In accordance with a pertinent contract.
Inappropriate	In violation of a pertinent contract.
Service	An actor, which on receipt of an application messages, may start an appropriate application operation. For example, a process which advertises an interface allowing defined RPCs to be invoked by a remote client.
Client	An actor which sends application messages to services.
Effect	The changes induced by the incomplete or complete processing of a set of procedures by an actor, which are observable by another contemporary or future actor, and which are made in conformance with a contract known to any such observer. This contract must state the countereffect of the effect, and is known as the countereffect contract. An effect is Completed when the change-inducing processing of the set of procedures is finished. [Need an indirect or consequential damage exclusion clause]
	<i>PRF - Sentence about countereffect contract doesn't fit well</i>
Ineffectual	Describes a set of procedures which has no effect.
Countereffect	An appropriate effect intended to counteract a prior effect.
Countereffect Contract	The contract which governs the relationship between the effect and the countereffect of a procedure. In the absence of any other overriding contracts the countereffect contract is the promise that

“The **Countereffect** will attempt so far as is possible to reverse or cancel the **Effect** such that an observer (on completion of the **Countereffect**) is unaware that the **Effect** ever occurred, but this attempt cannot be guaranteed to succeed”.

Cancel	Process a countereffect for the current effect of a set of procedures.
Confirm	Ensure that the effect of a set of procedures is completed.
Prepare	Ensure that of a set of procedures is capable of being successfully instructed to cancel or to confirm.
Outcome	A decision to either cancel or confirm.
Participant	A set of procedures which is capable of receiving instructions from a coordinator to prepare, cancel and confirm. A participant must also have a BTPA to which these instructions will be delivered, in the form of BTP messages. A participant is identified by a participant identifier.
Inferior Identifier	An identifier assigned to an Inferior which is unique within the scope of an Address-as-Inferior.
Atomic Business Transaction <i>or</i> Atom	A set of participants (which may have only one member), all of which will receive instructions that will result in a homogeneous outcome. (Transitively, a set of operations, whose effect is capable of countereffect.) An atom is identified by an atom identifier.
Atom Identifier	A globally unique identifier assigned to an atom. <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"><i>PRF – abs msgs define as unambiguous in scope of its address-as-superior, I think.</i></div>
Coordinator	An actor which decides the outcome of a single atom, and has a lifetime which is coincident with that of the atom. A coordinator can issue instructions to a participant to prepare, cancel and confirm. These instructions take the form of BTP messages. A coordinator is identified by its atom’s atom identifier. A coordinator must also have a BTPA to which participants can send BTP

messages.

Address-as-Superior	The address used to communicate with an actor playing the role of an Superior
Address-as-Composer	The address used to communicate with a Composer by an application actor that controls its resolution. The messages that might be sent to or received from this endpoint are undefined.
Address-as-Inferior	The address used to communicate with an actor playing the role of an Inferior.
Identity-as-Superior	The combination of Superior Identifier and Address-as-Superior of a given Superior.
Identity-as-Inferior	The combination of Inferior Identifier and Address-as-Inferior of a given Inferior.

4766